THE ST QUARTERLY

START

U.S.A. $14.95 CANADA $19.95
Disk Enclosed

Fall 1986 Volume 1, Number 2

FEATURES:

Play REVERSI
A Desktop Accessory

David Small explains the
Atari Disk Controller

Handy Reference:
Seven Word Processors
Over 90 Features Compared

Silicon Soothsaying
Tim Oren Predicts...

"Which C for Me?"
C Compilers Compared

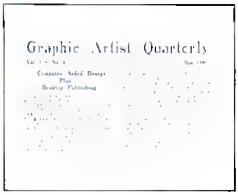
Tom Hudson's
Amiga Art on Your ST

Experiment with
Expert Systems

FujiBoink! Exposed

0 74470 17796 63

the Graphic Artist®



NOW ONLY

$199.95!

reduced from $495.00

All sample designs were designed and printed using the Graphic Artist

the Graphic Artist FONT EDITOR Now Sold Separately for $79.95

The Complete Professional

COMPUTER AIDED DESIGN & DESKTOP PUBLISHING

System for the Atari ST

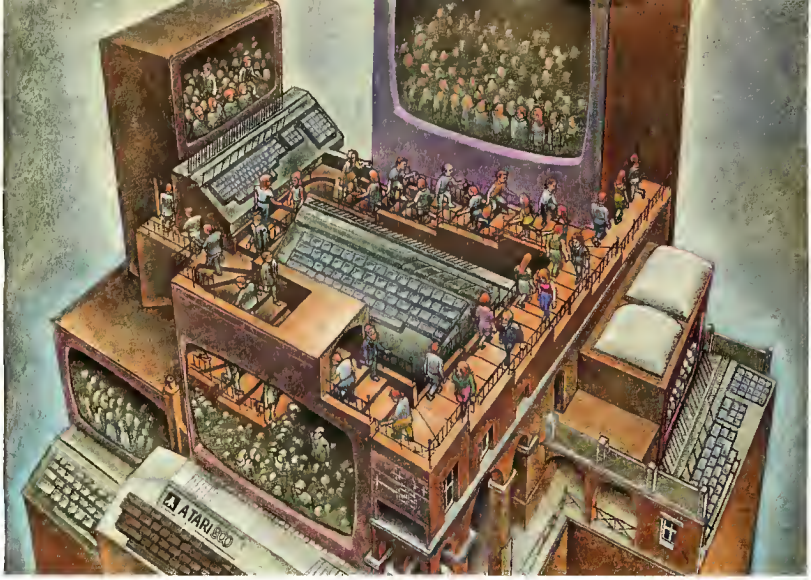
"The GRAPHIC ARTIST is one of the first programs that tries to take advantage of the tremendous power of the ST . . . the first ST program to use the printer to its highest resolution and the results are extremely impressive . . . The GRAPHIC ARTIST can indeed produce professional drawings, not available through EASY DRAW . . . quite an impressive product."

—Current Notes

PROGRESSIVE COMPUTER APPLICATIONS

2002 McAULIFFE DR. ROCKVILLE, MD 20851 (301) 340-8398

CIRCLE 092 ON READERS SERVICE CARD



USE THE BRAINS YOUR ATARI WASN'T BORN WITH.

Right at your fingertips in CompuServe's Atari® Forums.

Our Atari Forums involve thousands of Atari users worldwide. These forums show you just how easy and how much fun it is to get the most from your Atari computer.

The **Atari 8-bit Forum** provides the chance to talk with Atari enthusiasts on all levels. You'll learn all kinds of useful information from all kinds of interesting people. It's the leading national Atari Forum.

Devoted exclusively to users of the ST-series, the **Atari 16-bit Forum** provides programs, textfiles, assistance, product reviews, transcripts of online conferences and more.

The **Atari Developers Forum** is the authorized distribution area for all updates to Atari's registered software developer's kit for both the 8-bit and 16-bit Atari Computers. Members may access conferencing channels, data libraries, free programs and software.

Easy access to free software.

- *Download first-rate, non-commercial user-supported software and utility programs.*
- *Take advantage of CompuServe's inexpensive weeknight and weekend rates (when forums are most active, and standard online charges are just 10¢ per minute).*
- *Go online in most major metropolitan areas with a local phone call.*
- *Receive a* ***$25.00 Introductory Usage Credit*** *when you purchase your CompuServe Subscription Kit.*

Information you simply can't find anywhere else.

Use the Forum Message Board to exchange mail with fellow members. Join ongoing, real-time discussions in a Forum Conference with Atari luminaries like Sam Tramiel and Bill Wilkinson. Scan Forum Data Libraries for free software, documentation and contributions from Atari enthusiasts.

Enjoy other useful services too, like electronic editions of your favorite magazines, newsletters and articles, including ANTIC, Family Computing, OMNI Online, and Electronic Gamer.™

All you need is your Atari computer and a modem . . . or almost any other personal computer.

To buy your CompuServe Subscription Kit, see your nearest computer dealer. Suggested retail price is $39.95. To receive our free brochure, or to order direct, call 800-848-8199 (in Ohio, call 614-457-0802). If you're already a CompuServe subscriber, just type GO ATARI at any ! prompt and see what you've been missing.

CompuServe®

Information Services, P.O. Box 20212
5000 Arlington Centre Blvd., Columbus, Ohio 43220

800-848-8199

In Ohio, call 614-457-0802
An H&R Block Company

CIRCLE 001 ON READERS SERVICE CARD

FALL 1986

THE ST QUARTERLY

START™

VOLUME 1, NUMBER 2

CONTENTS

"Parsers! I hate parsers. . ."page 54

Step up to Expert Systems. page 22

Only a cartoon?. . . . . page 109

FEATURES:

Play Reversi On Your Desktop
A Desk Accessory Tutorial
Christophe Bonnet 12

The Amazing Mousetrap
Avoid Annoying Menu Drops
Daniel Moore & David Small 19

The AI Apprentice
Explore Expert Systems With XLISP
Christopher Chabris 22

Seven Word Processors Examined
Ian Chadwick 32

Swapping Art With Other Computers
IFF Standards For The ST
Tom Hudson 39

Parsers, Rooms, Objects and Trolls
The Adventurer's Tale
Alex Leavins 54

Probing the FDC
All About The Floppy Disk Controller
David Small 96

FujiBoink! Behind the Bit Planes
Find Out How
XANTH Park 109

DEPARTMENTS:

Procedures
Structured I/O
Complicated C Technique Simplified
Harry Koons 77

Perspectives
Silicon Soothsaying
Tim Oren 82

Developer's Notebook
Transportable GEM
IBM to ST
Mark Skapinker 90

REVIEWS:

Which C For Me?
START's First C Comparison
Arick Anders & Michael Bendio 62

Editorial 6

Dialog Box 7

Alert Box 11

START Chart 36

Disk Instructions 50

Clipboard 75

Authors 116

Resources 120

Advertising Information 122

Indicates programs included on START Disk. See order card at page 18A if you did not buy Disk Version.

ELECTRONIC ONE *

CALL (614) 864-9994 WRITE P.O. BOX 13428 COLUMBUS, OHIO 43213

ATARI 520 ST

COLOR SYSTEM **748.00**

BLACK & WHITE SYSTEM **468.00**

BOTH SYSTEMS:

- 360K DRIVE
- BUILT IN TOS
- 512 KEYBOARD
- SOFTWARE

EXTRA SGL DRIVE 59.99
w/purchase of system

ATARI 520 ST SOFTWARE

PERSONAL PASCAL	44.99
COPY II S.T.	26.99
VIP PROF	99.99
VIP LIGHT	74.99
MUSIC STUDIO	34.99
UNIVERSE II	47.99
PEGASUS	26.99
PRINTMASTER	26.99
P.M. ART GALLERY	17.99
TYPESETTER	22.99
ST TALK	13.99
SILENT SERVICE	24.99
SUN DOG	24.99
THE PAWN	26.99
BRIDGE	18.99
ROGUE	24.99
TEMPLE APSHAI	24.99
WINTER GAMES	24.99
MACRO ASSEMBLER	54.99
BRATACCUS	32.99
N-VISION	26.99
MAJOR MOTION	26.99
LEADER BOARD	24.99
HIPPO RAM DISK	24.99

ATARI 520 ST HARDWARE

SF 354 DISK DRIVE	119.99
SF 314 D/S D/D	199.99
SHD 204 ATARI HARD DRIVE	649.99
SM 124 12" B/W HIGH RES MONITOR	119.99
12" MED RES RGB MONITOR	279.99

HOW TO ORDER: CASHIER CHECK, MONEY ORDER, MASTERCARD*, or VISA* (Add 4% for charge cards) . . NO PERSONAL CHECKS . . . NO C.O.D.'s . . SHIPPED U.P.S. PRICES SUBJECT TO CHANGE

SHIPPING: Prompt one day shipping on in-stock merchandise. Ohio residents add 5.5% sales tax. Add $3.00 on all orders under $100.00. Add $5.00 on all orders over $100.00

INTERNATIONAL: Actual freight charge on all orders outside the continental United States including A.P.O.

CALL OR WRITE FOR FREE CATALOG

CIRCLE 020 ON READER SERVICE CARD

Solapak ST

Lean & Mean!

Print Spooler/Ram Disk

Now with the ***"Solapak ST"*** print spooler you can:

- print up to 8 files with separate printing qualities at one time.
- Configure to any parallel printer with 32 user definable options.
- Adjust line spacing, format your output, vary the print speed, pause/restart printing and much, much more.

The ***"Solapak ST"*** Ram Disk lets you:

- configure up to 1 Mb and 128 files.
- save time with a disk transfer rate of over 10 million bits/second.

Includes user adjustable screen saver to help prevent image "burn-in".

"Solapak ST" is easy to use and the fastest, most versatile and memory efficient professional print spooler and ram disk available for the **ATARI ST.**

"Solapak ST" demo at your ST retailer or on BBS's nationwide.

$39.95 Not copy protected

Action Software - 69 Clementina St. Dept. SR
San Francisco, CA 94105 - (415) 974-6638

Mail orders: please include $3.00 S&H and Ca. residents add $2.60 sales tax. VISA, MC OK.

CIRCLE 003 ON READER SERVICE CARD

STation

Integrate Your ATARI ST

- An integrated workstation for homes and offices
- One power switch.*
- Save valuable desk-top space
- Protects disk drives and monitor from magnetic fields
- The Station is vented to keep 520ST cool.
- Holds 2 floppy disk drives.
- Increased portability.
- The 520ST slides in and out easily from under the chassis, with adequate space for a mouse & joystick connector.
- With optional surge surpressor

DEALER INQUIRIES WELCOME!

ATARI & 520ST are registered trademarks of ATARI Corp.

SOUTHERN TECHNOLOGIES, INC.
3208 Beltline, Suite 210
Dallas, Texas 75234
(214) 247-7373
1-800-647-7741

CIRCLE 113 ON READER SERVICE CARD

START THE ST QUARTERLY

PUBLISHER
James Capparell

EDITORIAL
DeWitt Robbeloth, Executive Editor
Jack Powell, Associate Editor
Matt Loveless, Technical Editor
Patrick Bass, Technical Advisor

CREATIVE SERVICES
Marni Tapscott, Art Director
Kyle Houbolt, Production Manager
Deborah Onodera and Katherine Murphy, Ad Production Coordinators
Jaciow Design, Cover Design
Contributing Artists: James E. Dowlen, James Endicott, Jim Warner, Kyle Houbolt, Ward Schumaker, Mary Rhomberg Lavery, John Hersey.

ADVERTISING SALES
John Taggert, Director
(Please see Advertising Index, page 122).

CIRCULATION
Margot Olmstead, Director
Dixie Nicholas, Subscription Coordinator

MARKETING
Jon Loveless, Vice President; Gary Yost, Director; Lisa Wehrer, Retail Sales Manager; Daniel Borrett and Ken Warner, Dealer Sales; Eric Gupton, Customer Service Coordinator; Scot Tumlin, Technical Support; Michael Sandstrom, Customer Relations.

General Offices, Catalog Customer Service, and Subscription Customer Service:
(415) 957-0886
8 a.m. to 1 p.m. Pacific Standard Time
START, 524 Second Street,
San Francisco, CA 94107
Credit Card Subscriptions & Catalog Orders:
(800) 443-D1OD ext. 133
(Continental U.S. & Hawaii)

Fall 1986, Volume 1 Number 2
START, The ST Quarterly, is published four times per year by Antic Publishing. Editorial offices are located at 524 Second Street, San Francisco, CA 94107. Application to mail at second-class postage rates is pending at San Francisco, CA.

POSTMASTER: Send address change to START, 524 Second Street, San Francisco, CA 94107.

SUBMISSION INFORMATION
START welcomes submissions. Please include both hard copy printouts of articles and program listings as well as disk files on ST compatible disks. Media will be returned if self-addressed, stamped mailer is supplied. START assumes no responsibility for unsolicited editorial materials.

START is an independent periodical not affiliated in any way with Atari Corp. ATARI is a trademark of the Atari Corp. All references to Atari products are trademarked and should be so noted.

START is a registered trademark of Antic Publishing, Inc.
An Information Technology Campony.
Copyright © 1986 by Antic Publishing
All Rights Reserved. Printed in USA.

EDITORIAL

You are now holding the second issue of *START—The ST Quarterly*, the most professional and complete magazine devoted to the Atari ST computer. Our first issue sold out, although there may be a few copies in inventory for those of you who want the complete set.

Finally the Atari is being recognized as a competitive machine. Advertisers familiar to readers of MacWorld and PC have been approaching us. But what of the most important ingredient in a successful magazine introduction, the reader? Look at the letters in our Dialog Box. You readers raised some interesting and valuable questions. Let me answer those regarding content, subscriptions and frequency.

We had responses ranging from "great," and "fantastic," to "I'm disappointed," and "over my head." My editorial position is, if you own an ST, then START belongs in your mailbox. I agree, our first issue had some very difficult material. Those of you who felt in over your heads should keep reading because we intend to bring you comprehensive buying guides, technical tips that simply require your machine to be plugged in, and more power-user material like how to get the most out of your spreadsheets, communications software, and languages.

Remember, the disk is included, so no typing. Just read and enjoy—let the computer do the rest. For you hacker types and developers, I think we've already got you covered. You probably noticed the high-quality writers we've already signed up, and you can count on more.

I often tell our dedicated staff, we should address three groups in START. For those of you who are still deciding, we'll show you why the ST is a good buy. For those of you who are new owners, we'll help you feel comfortable with this powerful system, and help you choose software. And, for those of you already familiar with the system, rest assured we will bring you the most complete and useful information, programs and tutorials.

All of this is easy to say, and difficult to do. After all, we're only two issues old. We'll make some mistakes, but a good, powerful, useful magazine takes some time to mature.

Many of you know we offer a $59.95 one-year subscription. You get four issues of START, each including a 3.5-inch disk, plus 12 monthly issues of Antic—the Atari Resource. Why Antic?

Our reasoning went something like this: in Antic we have already built up the number one circulation, sales and advertising base for Atari computing. We already have withstood the test of time—four and a half years—in an unforgiving market. Last year we started the ST Resource section in Antic, and it grew. Non ST readers worried that the ST would take over the whole magazine, so we promised we would limit ST coverage to no more than 40 percent of Antic's editorial pages. Now this coverage is necessarily less comprehensive than START's and the articles are shorter and easier, but it's still 300 to 400 extra pages of ST material every year, and probably just right for many of you who are learning the ST.

As your skill increases, and you demand more thorough material, then there you have it, another quarterly issue of START, more difficult but much more to chew on.

EDITORIAL

Well, it still seems right to me, so I'm going to ride with this package awhile. If we deliver you START and Antic, and leave you excited, filled with ideas, more confident, and looking forward to another issue, then we will have done our job, delivered you good value for your money.

As for frequency, I think as we better understand our changing market, you can expect *START–The ST Quarterly* to become at least bimonthly.

Jim Capparell

James Capparell
Publisher

Answer: *1040ST*™

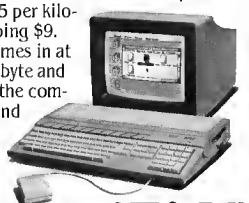
Question: *Which computer is the first in the world to give you 1 Megabyte of power for under $1,000?*

The 1040ST is a major breakthrough in personal computers. Indeed, it's the world's first computer with an original list price that represents less than $1 per kilobyte.

To give you an idea of what an extraordinary accomplishment that is, let's look at the price-per-kilobyte figures for some well-known competitors.

The Macintosh,™ for example, comes in at over $4 per kilobyte, the Amiga™ is over $5 per kilobyte and the PC AT™ is a whopping $9.

In contrast, the 1040ST comes in at an incredible 98 cents per kilobyte and a total price of just $999.95 for the complete system: CPU, disk drive and high-resolution monochrome monitor.



ATARI and 1040ST are trademarks of Atari Corp.
Amiga is a trademark of Commodore-Amiga, Inc
AT is a trademark of International Business Machines, Inc.
Macintosh is a trademark licensed to Apple Computer, Inc
© 1986 Atari Corp.

CIRCLE 005 ON READERS SERVICE CARD

ATARI®

DIALOG BOX

INTRODUCING STING

Congratulations on your premiere issue. It looks great, and I'm impressed by the talent you have writing for you and the diversity of the articles. David Small's "Voodoo Computing" started a small debate among my programmer friends, but most of us agreed with his maxims.

We have just formed an ST users group in the Peoria, Illinois area. We call it STING–for ST INformation Group–and our only rule so far is "no pirated software, either for demonstration or for trading." We'd like to hear from others who want to exchange newsletters.

David Stambaugh
109 Florida Avenue
Washington, IL 61571

Thanks for the encouragement. START will publish user group information as we get it; see Resources this issue.

NO PROGRAMMER

After reading the first issue of START I agree with Mr. Capparell's editorial that he has published a magazine for those who want "high-end information that requires grounding in computer fundamentals." But what about the rest of us?

I'm a 37 year old teacher, and the ST is my first computer. I bought a few games and applications for it, and I LOVE IT. My problem is that your articles make me feel like a fourth grader who has wandered into a college physics class. I'm not a programmer and don't have the time to become one.

I know you can't please everyone, but I think I'm representative of many new ST buyers–enthusiastic and ready to spend on hardware and software (if only teachers made more money).

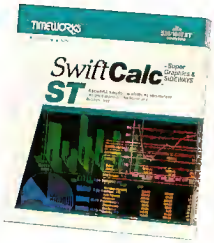
I think it's great to publish a magazine for the ST only, and it's great to show Atarians how to use their computers in new exciting ways, but if you want me as a subscriber, you'll have to take things a little slower.

Larry Jones
Pocatello, ID

We acknowledge your concern and its echoes from other readers. See our Editorial this issue, and remember, a subscription to START includes 12 monthly issues of Antic magazine, with substantial ST information and programs for the beginner. Also, you don't have to be a programmer to run and ▶

IF YOU CAN FIND A BETTER PROGRAM WE'LL BUY IT FOR YOU!

Full GEM interfacing, convenient Quick Keys, and many other unique features of our own.



WORD WRITER ST with Spell Checker

A full-featured, professional word processing system for home and business use. You get:

- **A Continuous Spell Checker** that identifies misspelled words as *you* are *typing* your document.
- **An additional 85,000 word, built-in Spell Checker** that checks your entire document – at your command!
- **On-Screen** underlining, *italics* and **boldface** – as you write.
- **An Outline Processor** that quickly organizes notes, facts, and ideas in convenient outline format.
- Every other feature you'll need for everyday word processing, plus most of the features found in more expensive programs.

DATA MANAGER ST with Report Writer

A complete general information Storage and Retrieval System with report writing, business graphics, and label making capabilities. Plus, you get:

- **Flexibility** that allows you to modify your data base structure by changing the size or position of fields of information – at any time.
- **A complete Report Writer** that generates customized data reports. You specify the title, location, and sequence of each column.
- **An extensive Business Graphics Package** with pie charts, bar charts, line plots, point plots, hi-lo-close stock price plots, and more.
- **A Label Maker** that prints your important information onto all types of labels.

SWIFTCALC ST with Sideways

A state-of-the-art spreadsheet for home and business use. You get:

- **8192 rows and 256 columns provide up to 2,048,000 cells** (locations) in which to place information.
- **Super Graphics** graphically display and print out business information using pie charts, bar charts, line plots, point plots, hi-lo-close stock price plots, and more.
- **Windows** allow you to work on two sections of your spreadsheet simultaneously.
- **Sideways** lets you print all columns on one continuous sheet . . . *sideways*.
- **Help Screens,** help you use the program – without referring to your manual.
- **Formatting** – Choice of five formats: Decimal (up to 9 places); Graphics; Exponential Notation; $ Sign & Commas.

With Timeworks you get more than software. . .

You Get Our Customer Technical Support Team – free to all registered users.

With Timeworks you get our Money Back Guarantee*
If you can find anything that works better for you – and it's available – we'll buy it for you. Details inside every Timeworks package.**

You Get Our Liberal Upgrade and Exchange Policy – Details are inside every Timeworks package.

TIMEWORKS®

More power for your dollar

Timeworks, Inc. 444 Lake Cook Road, Deerfield, Illinois 60015
312-948-9200

For the Atari 520/1040 Computers.***

Suggested Retail List Price:
$89.95 each

Available now at your favorite dealer, or call Timeworks.

TO ORDER CALL:
1-800-535-9497

These Programs INTERFACE with Each Other

© 1986 **Timeworks**, Inc. All rights reserved.

* Offer expires 90 days after date of original purchase

*** Registered trademark of Atari Corp.

CIRCLE 123 ON READERS SERVICE CARD

enjoy the programs on the START disk.

WANTS THE DISK

I picked up my first copy of START and was disappointed to find it had no disk. The vendor said because it came both ways it was cheaper for him to sell it without the disk. How do I get the disk?

Joseph Santorelli
Rome, NY

START is a magazine/disk combination, not intended for separate sale. In order to get established we are making some nondisk copies available, especially to newsstands. Many Walden and Dalton bookstores, and retail computer stores sell the disk version, and START disks are also available from Antic Publishing for $10.95 plus handling (see order form in this issue).

DOESN'T WANT DISK

Just picked up a nondisk copy of START. It is amazingly good, but I'm desperately broke, so I don't own an ST yet. I hope the nondisk version will continue to be available for those of us who don't yet have the machine.

Nicholas Bodley
New York, NY

We will continue to make nondisk copies available for retail sale, but subscriptions copies all contain disks.

GOBBLEDYGOOK

I bought a 1040 ST, and I'm in love. I have no programming knowledge, nor do I wish to acquire any. My computer provides an easy and enjoyable vehicle for using many business, hobby and entertainment programs that are elaborate, and yet a snap to run.

Imagine my excitement when I found START, a magazine devoted entirely to the ST. Also imagine after I dropped $15 to find the same old gobbledygook I have worked hard at avoiding. I'd like more software reviews in depth, and more information about peripherals. Please find some way to help those of us who spend money happily on our ST, but merely want to use it.

Betsy Dobrick
Coral Springs, FL

We plan to always have enough general information and useful programs to make your purchase worthwhile. See Antic magazine for reviews and news of ST products. Also see START's new Clipboard for hints and tips the nonprogrammer can use.

Answer: 1040ST™

Question: Which computer was specially designed for people who hate to wait?

Let's face it, any time you spend waiting on a computer is time wasted. That's why Atari® built the 1040ST with a sizzling clock speed of 8 MHz.

And with 1024K bytes of Random Access Memory, the ST™ gives you an incredible combination of power and speed. (The PC AT,™ for example, has 512K of memory.)

So you'll spend time working on your ST, instead of waiting on it.

In addition, the 1040ST costs an amazingly low $999.95, which makes it the first computer in the world to deliver 1 Megabyte of memory for under $1,000. (The PC AT costs about $4,500.)

So if you haven't checked out the ST yet, what are you waiting for?

ATARI, ST, and 1040ST are trademarks of Atari Corp.
AT is a trademark of International Business Machines, Inc.
© 1986 Atari Corp

ATARI®

BOUNCING ALONG

Having seen Amiga's Bouncing Ball program and also the Fuji symbol variation run on the ST, I have a burning desire to know how it was done. I'd love to create impressive graphics like these.

R.G. Summers
Renton, WA

Your wish is our command. See FujiBoink! in this issue, also source code for it on your START disk.

ABSORBING START

Yesterday I got the first issue of START. There's a lot in it to absorb. I may be absorbing until the next issue is published. What a great start!

But I must say I was sorely disappointed not to find a word processor program reviewed. I have 1ST Word, but abandoned that as soon as I found Regent Word, which I fully expect to use for a long time to come.

I hope to see carefully researched reviews in START that will help us readers choose useful programs.

I want us all to succeed, to grow and get better. I thank you and anticipate a satisfying future.

Joseph C.P. Alvarez
Omaha, NE

The information on word processors in this issue's START CHART should wet your whistle. ▶

THE NEVERENDING STORY

Save the world of "Fantasia" in this thrilling adventure. As the hero Atreyu, you face the trials and terrors of the "All-Consuming Nothing." Enter the quest in this richly illustrated and highly imaginative graphics-text adventure game.

Datasoft® ...We Challenge You

19808 Nordhoff Place, Chatsworth, CA 91311 (818) 886-5922

CIRCLE 015 ON READERS SERVICE CARD

© Neue Constantin Filmproduktion GmbH 1984. Datasoft is a registered trademark of IntelliCreations, Inc. © 1986 IntelliCreations, Inc.

DIALOG BOX

FAST SATISFACTION

Please run some articles that give comparative results for different hardware and software. How about a chart showing compile and execution times for a number of C compilers for the ST?

Brian Cole
CompuServe 73107,1756

You must have read our minds. See C comparison article and START CHART on word processors in this issue.

USE US OR LOSE US

I'm a long time Atari user (with the emphasis on user), and Antic reader (I've still got every one). I've stuck through some thick and much thin, but I never became a programmer. Every man has his limitations, and plumbing and programming are just two of mine.

Don't get me wrong; I want to read programming articles to understand the logic of what's going on, but more than that I want fundamental instruction, with tips and techniques about the machine's many features. I want useful utilities (like Tom Hudson's printer drivers), and answers to my questions (like how *do* you use the mail merge in ST Writer?).

Your first editorial referred to us "experienced users," but I'm not sure your content does. I realize the machine is new, as is your publishing effort. START looks very good, seems well written, but needs time to mature.

I'll wait and see. Thanks for the magazine, keep up the good work, and remember that many, if not most, of us want to be users, not programmers.

Jim Cummings
Pennsauken, NJ

Roger, Pennsauken. We read you loud and clear. Observing new vector. Over and out.

Answer: *1040ST*™

Question: *Which computer builds in multiple features instead of hidden costs?*

It seems that a lot of our competitors design stripped down computers, and then charge extra for every feature and upgrade you add.

Atari® doesn't do that, because we believe the features and level of performance you want should be built in to begin with.

That's why the 1040ST gives you a full Megabyte of memory. While the competition only gives you the chance to spend big dollars trying to improve their memories.

Another trick they use is to make sure their interfaces don't meet industry standards, so you're locked into their system. In contrast, the ST™ uses standard interfaces across the board, such as the RS-232C port for serial modem communications and the parallel interface for an industry standard printer.

Of course, the ST's best built-in is the price, which is an incredible $999.95!

ATARI, ST, and 1040ST are trademarks of Atari Corp.
©1986 Atari Corp.

ATARI®

ALERT BOX

Charles Roberson of Vienna, VA noticed two errors in Dan Matejka's "AL and C routines" from the first issue of START. On page 65, an erroneous semicolon appeared:

```
*--d;=*s;
```

Unless the first semicolon is removed, a compile-time error will ensue. The second error concerns the MC68000 AL code presented as output from the Alcyon compiler on page 66. A **bra** to the second **move** inside **moveloop** was inadvertently left out.

WHERE'S MY DISK?

START is a magazine with its programs on disk. Normally the disk is bound into the magazine and sells on the newsstands for $14.95.

But some of you ST enthusiasts want to read START first, so we have provided a limited number of copies without disk for $4.00 each.

If this is your situation, you can complete your copy of this issue of START by ordering the companion disk direct from us, for $10.95 plus $2.00 shipping and handling. See the handy order form.

PLAY REVERSI ON YOUR DESKTOP

A GEM TUTORIAL AND A GREAT GAME

by Christophe Bonnet

This desk accessory version of Reversi (or Othello) is the perfect executive toy far thase lengthy spreadsheet sessions. French pra-grammer Christaphe Bonnet uses this classic camputer game to demonstrate haw ta create a desk ac-cessory.

MARY RHOMBERG LAVERY

Although the 68000 processor in the Atari ST is fully capable of running several programs simultaneously, the current version of GEM does not support such multi-tasking. It *does*, however, support "multi-programs." When a program is designed as a desk accessory and properly installed, it can share Desktop space with up to six other GEM programs. During operation of any one of these programs, the others cease foreground activity. Thus, desk accessories do not provide true multi-tasking, though they can be a convenience.

Desk accessories, such as the familiar Control Panel, are programs that have been designed and compiled specifically to be accessories. When placed on the "boot" disk, GEM recognizes these files by their .ACC extender and loads them into a safe place in RAM. Titles for each accessory will appear in the "Desk" drop-down panel on the Menu Bar and will therefore be accessible from within any GEM program displaying the Menu Bar. The titles are created by the accessory programs and are not necessarily the same as the program filenames. More than one desk accessory may be included within a program file, but GEM supports a total of six desk accessories, no matter what the number of program files.

Many desk accessories are available through the public domain, such as calculators, clocks, and slide puzzles. Commercial desk accessories are also beginning to appear. The advantages as well as the limitations of a program designed as a desk accessory suggest certain creative directions. In this case, I chose the ubiquitous computer game, Reversi. It has appeared in various forms on practically every computer from mainframe to micro. As an ST desk accessory, it provides a refreshing spot of strategic entertainment during long hours at the computer.

REVERSI RULES

Reversi is a popular programming challenge. The rules of the game are very simple, but developing a good strategy takes skill. Reversi is found on many computers because the game mechanics are easy to understand and the graphics requirements minimal. (The tokens on mainframe versions are usually made up of X's and O's.)

In our version of Reversi, the human player always takes the black token and the ST computer the white. Each side alternately places one token of its own color on an unoccupied square, so that the following rule is respected:

The token must be placed so that an imaginary straight line, drawn from that token to any other token of matching color will cut through one or more tokens of the opposing color. (See *Figure 1.*)

Following a move, all opposing "sandwiched" tokens flip to the active player's color. So, if black's token was placed on the bottom "?" in *Figure 1*, the board will change as in *Figure 2*.

With each move, you must take at least one opposite-color token between two of yours or "pass" for that move. The game continues with each side alternately placing tokens and capturing pieces until all squares are occupied, or no legal move is possible. The winner is the player with the most tokens of his or her color.

USING THE REVERSI DESK ACCESSORY

The Reversi desk accessory file is found within the REVERSI.STQ folder on your START disk. Since it is a desk accessory, copy the file REVERSI.ACC to a boot disk, then turn on your ST with this disk in the main drive. Now, click on the REVERSI label that should appear in the Desk drop-down menu, and the Reversi window will appear in the middle of your screen with two tokens for each side preset in the middle of the board.

As with most desk accessories, you may click in the title area and drag the Reversi window anywhere on the Desktop. Though you may not size the window, you may close it by clicking on the upper-left corner box. Unless you reboot, Reversi will remember your last position and you may continue your game as you left off. The current score is on the right side of the window.

You are black and you move first. Point the mouse cursor where you want to place your token and click the left button. If your choice is legal, the computer will place your token and flip the appropriate opposite tokens. It will then take its own turn. Continue in this way until all squares are filled or there are no more moves. After the winner is declared, you may play again by clicking anywhere within the window. If you can't make a legal move, resign your turn by pressing any key on your ST.

THE PROGRAM

If you plan on compiling Reversi from the source code on the START disk, please see the Disk Instructions page in this issue for details regarding the files related to this article. The program was written with Alcyon C from the Atari Developer's Toolkit. Other C's may require some modification.

A small part of Reversi code is taken from ACCSKEL.C, which stands for "accessory skeleton," and may be found in the Developer's Toolkit. Those parts are indicated both in the listing and in the following analysis. ACCSKEL.C is ex- ▶

actly what it sounds like: a skeleton structure to make your own code into a desk accessory.

Looking at REVERSI.C on your START disk (or on a printout if you insist on doing it the easy way), you can see that the only external file needed is GEMDEFS.H. The first cluster of **#define** statements makes the code easier to read, and the second group takes care of the only XBIOS calls we use, thereby eliminating the need to **#include** OSBIND.H. Following this are a number of global variables including several from the original ACCSKEL.C.

The second declaration group contains variables used by the game subroutines. The array **b_grid**[][] will contain the values of each game square; positive values represent good squares and negative values bad ones. (A few quick games will quickly demonstrate why some squares are "bad".) These values will change during the course of a game, but **b_init**[][] is the same array holding the starting values for the squares before the first move occurs. **c_grid**[][] is an array of codes showing the contents of each square: 1=empty square, 2=player, 3=computer, and 0=border (not a valid square).

Notice, among the other variables, **phase** and **adv**. These are used in some procedures to tell the computer which side is playing. A value of two indicates human and three computer; **phase** represents the current player and **adv** represents the opponent during a move.

Two standard GEM routines, **open_vwork( )** and **open_window( )**, open the virtual screen workstation and open a window of specific size and attributes for the Reversi playfield. Both routines are from the ACCSKEL.C source code.

Following these two procedures is the **main( )** entry point of the program. This routine is called by the GEM shell immediately before displaying the Desktop. In this section is **appl_init( )** which should be the first function in any GEM program. The **menu_register( )** function places the desk accessory's menu item string, "Reversi," on the Desk menu. Also in **main( )** are **screen( )** and **init( )**, initialization calls for the game itself, which will be explained later in this article. Following this is **multi( )**, which contains our "psuedo-multitasking" routine.

THE ROUTINES

The portion of the program which enables the desk accessory to interact with any GEM application is **multi( )**. The AES function **evnt_multi( )** continually checks for GEM calls until you turn off the power or press the reset button. The purpose of **evnt_multi( )** is to wait for specified events chosen by the programmer. When the event occurs, the operating system interrupts the current application and begins the appropriate program—or desk accessory. The specified GEM events for Reversi are **MU_MESAG**, **MU_BUTTON**, **MU_KEYBD**, **MU_M1**, and **MU_M2**.

MU_MESAG is for specific window events (sizing, moving, closing, etc.), **MU_BUTTON** is the mouse button state (down or up), **MU_KEYBD** represents keyboard input, and **MU_M1** and **MU_M2** are mouse position events. Mouse position events occur when the mouse cursor enters or leaves a specific rectangle, in this case, when the cursor exits or enters the Reversi window. With this information, we can change the cursor form to a pointed hand if it is in the window.

Throughout all of **multi( )**, the program dispatches the various calls in order to execute various tasks (window moving, cursor modification, accessory call by Menu bar, etc.). The **multi( )** function is normally found in all GEM programs using the Menu Bar.

baard( )

Following a window updating message from GEM, or when you open the Reversi window, **board( )** is called upon to draw the window (game board, tokens, score, etc.). Most of the functions used in **board( )** are from GEM VDI (line drawing, ellipses). The drawing parameter variables are adjusted depending upon resolution so the game will work in any mode.

set_clip( )

ACCSKEL.C provides both **set_clip( )** and **do_redraw( )**. The **set_clip( )** routine uses the size parameters of a given rectangle to "clip" any screen graphics which are drawn beyond this rectangle; **do_redraw** redraws any rectangle portions which have been "clipped" from the Reversi window by other windows. For example, when you drag a window on top of the Reversi one, then remove it, parts of the Reversi window must be redrawn. This function finds the parts and processes the subsequent drawing.

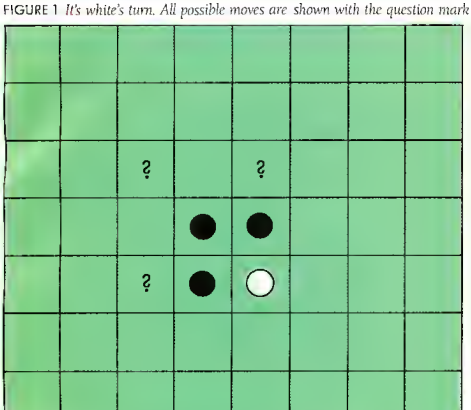
player( )

During the player's turn, **player( )** is called. It first converts the mouse X and Y coordinates (captured by **evnt_multi( )**) into grid square coordinates ranging from zero to eight. If the cursor is really pointing on a square, player subroutine **sub_1( )** is called.

atari( )

The **atari( )** procedure contains the core game strategy for Reversi. The program first scans the **c_grid** array for empty squares, then checks their playability with **verify( )**, which returns either zero (not a valid move), or the number of opposing tokens which would be flipped by this move. Then **atari( )** assigns a specific value to this square, depending on number of returned tokens and the strategic position of the square:

FIGURE 1 *It's white's turn. All possible moves are shown with the question mark (?)*



turn()

One complete game turn includes both the human and the computer and is handled by **turn()**, which is called from the **multi()** procedure whenever the player presses a key or clicks in the window. First, **turn()** checks the **end** flag to see if the game is over. If so, **init()** is called to redraw the game board and reset the squares to their starting values. Otherwise, **player()** is activated. (See above.)

If the player's move is valid, **atari()** determines the computer's decision. At the same time, the program checks for an end-of-game situation (all squares played: **score[2]+score[3]=64**, or both sides could not play anywhere: **pass==2**).

Lastly, **turn()** uses the VDI function, **wind_set()** to add the player prompt: "**Your move. . .**\0". ▶

```
Value=(b_grid[x][y] * 3)+flip
```

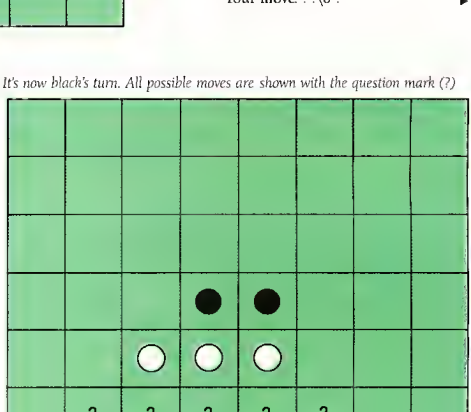
Because the number of returned tokens is more important near the end of the game, this value is affected by the course of the game. (This is a personal choice. You can modify the scoring to fit your own strategy.)

The computer then chooses the best square value and plays it: **change(v,w)** If there is no playable square, it resigns for this turn: **(point==-99)**.

display()

To update the score on the right side of the window, **display()** uses VDI functions. Further down in the listing, **rectangle()** draws a filled rectangle of a chosen color, using the VDI **v_bar()** function. Here, it is used to erase the previous score before writing a new one. Much of **display()** is devoted to converting the score numbers to strings for the VDI **v_gtext** function.

FIGURE 2 *It's now black's turn. All possible moves are shown with the question mark (?)*



Looking for a desk accessory? There are plenty available, and the supply is growing. Here is just a partial list of ST desk accessories which may be found commercially, or free on CompuServe or the Atari BBS.

Michtron leads the pack in desk accessories which include:
ALT–Alt-key macros make your life easier. Up to 36 programmable combinations ($29.95)
Calendar–A complete appointment and reminder calendar, including alarm clock. ($29.95)
Cornerman–This monster multi-function desk accessory includes calculator, notepad, clock, phone dialer and directory, and a game thrown in for fun. ($49.95)
Michtron, 576 S. Telegraph, Pontiac, MI 48053, (313) 334-5700.
CIRCLE 218 ON READER SERVICE CARD

Batteries Included's Thunder! ($39.95) is a real-time spelling checker. See Mark Skapinker's article in this issue for more on this program. Batteries Included, 30 Mural Street, Richmond Hill, Ontario L4B 1B5, Canada, (416) 881-9941.
CIRCLE 219 ON READER SERVICE CARD

Apex Resources is distributing a British product called Rhythm ($34.95), which is a desk accessory calculator in the format of a small spreadsheet. Apex Resources, 17 St. Mary's Court, Brookline, MA 02146, (617) 232-9686.
CIRCLE 220 ON READER SERVICE CARD

The Catalog will soon release Crystal ($24.95), an accessory that lets you access all GEM Desktop commands (including some never implemented by Atari) while within other GEM programs. The Catalog, 524 Second Street, San Francisco, CA 94107, (800) 443-0100 ext. 133.
CIRCLE 230 ON READER SERVICE CARD

COMPUSERVE ACCESSORIES

Get on line, type GO ATARI16, then browse through the data libraries. We found most of these in DL3, though some are also from DL0 and DL1.

PUZZLE.ACC–A tile-slide puzzle.
BICAL2.ACC–A three-resolution calculator.
BLUE.ACC–Turns your Desktop blue without needing the Control Panel.
CALC.ACC–Another calculator; take your pick.
CALEND.ACC–How about a desk calendar?
CRABS.ACC–Nasty little critters from Alex Leavins.
DIRPRI.ACC–Prints disk directory in condensed mode on Gemini 10X.
DMPNEC.ACC–NEC printer screen dump written in Personal Pascal.
FIXPNL.ACC–A fixed Control Panel that maintains date and time on reset.
RAM.ACC–Tells you how much free RAM is left
SCRSAV.ACC–Turns off your screen if you leave your CRT.
MITES.ACC–April fool's day joke. Similar to CRABS.ACC.
RAMDSK.ACC–This is just one of many RAMdisks in the libraries.

ATARI BBS–(408) 745-5308 will put you online with Atari. Go to SIG #8 for ST goodies. Among the more than 200 programs to download are the following accessories (many accessories listed above are also in the Atari BBS):
CLOCKA.ACC–A graphic analog clock for your Desk.
DIGICLOK.ACC–Or, how about a digital clock?
97KRAM.ACX–Like CompuServe, there are a lot of RAMdisks here.
SNAPSHOT.ACX–Saves GEM screen in DEGAS format.
EMULATOR.ACC–Updated VT52 emulator.
CLI.ACC–Command Line Interpreter.
PALLET.ACC–Adjust your color palette without the Control Panel.

sub_1() and sub_2()

The **sub_1()** function checks if the player's chosen square is empty before going on to **sub_2()**, which uses **verify()** (see below) to get a count on the number of opposing tokens before flipping their colors.

verify()

The **verify()** function scans all eight directions from a specified square for valid, opposing tokens. The variable for the number of opposing tokens in one direction is called **flap**, while **flop** (the number returned by **verify()**), is the total number of opposing tokens found in all eight directions.

change()

The **change()** procedure is the most complex one in Reversi. It draws a token on the board window, then updates the necessary variables to reflect the changes.

After activating a clipping rectangle within the window, and choosing some pleasing values for the drawing mode, the VDI **v_ellipse()** function draws the played token on the screen.

As mentioned, all graphics within Reversi are designed to look the same regardless of resolution. To correct for vertical deformation in medium resolution, the **−x** adjustment values are doubled when in this mode. Similarly, all high-resolution drawing values are adjusted to be twice as big as low ones to create the same size board in all three modes.

Next, all directions from the played square are scanned for opposing tokens. The tokens are placed into the variables (**c_grid**[]**[1**=**phase**) and onto the screen (**v_ellipse**). For each returned token, the score of both sides is modified accordingly in **score**[**phase**]++ and **score**[**advl**]−−.

Finally, **wind_update** is recalled to signal the end of window modification.

modify()

After a few games, you will notice that different squares have different strategic value during the course of play; in fact, the overall value of each square closely depends upon the status of its adjacent squares. If we want an acceptably challenging game, we must adjust the values of the squares appropriately. That is the purpose of **modify()**, which is called after each move. Only some squares are examined, to save memory space (and because I didn't want to spend months on this problem). I urge you to customize this routine for your own use. The best game results will come from improving this portion of the program.

stop()

As indicated by its name, **stop()** is called when no more moves can occur. A brief diagnostic message appears in the window using the **wind_set()**.

init()

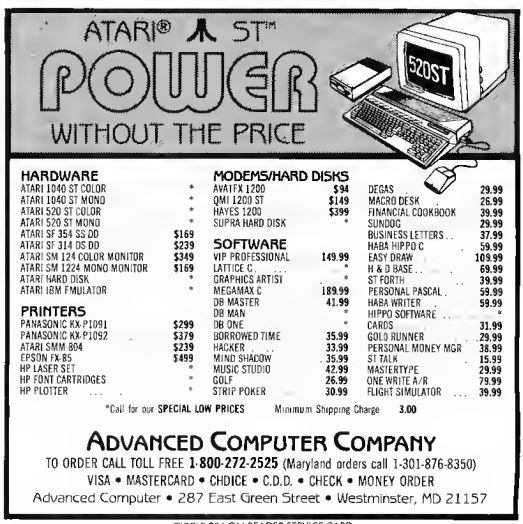
The **init()** function initializes all the **b_grid**[][1] array variables (importance of each square) with those in **b_init**[][]. Also, the **c_grid**[][1] array (content of the squares) is initialized with a one (empty) in each square, a zero (border) in all peripheral rows and a two (player) and three (computer) in central squares.

wait()

The delay loop **wait()** is necessary. Without it, the computer responds too fast for the player to see where the token was played.

screen()

During desk accessory initialization, **screen()** is called once by **main()**. It checks the current screen resolution, with **getrez()**, then allocates the appropriate values to the drawing variables used for window, board, tokens and text drawing. ■



CIRCLE 004 ON READER SERVICE CARD

START

THE ST QUARTERLY

PREMIERE ISSUE

See START Risk Offer Inside

Supply Limited!

U.S.A. $4.00
CANADA $4.9

Summer 1986 Volume 1, Number 1

THE RIGHT START!

Subscribe to START, the ST Quorterly, while there is still time to receive the PREMIERE ISSUE.

Quontities ore limited, however, so pleose don't deloy!

Remember, in oddition to 4 issues of START ond four 3½ inch progrom disks, you olso receive—FREE—o full yeor (12 issues) of Antic, which feotures the ST Resource every month.

So—for complete ST coveroge—look for Antic ond START ond be sure thot you hove oll the informotion you need to get the most out of your new ST.

Mail the attached card today!

THE AMAZING MOUSETRAP

ANNOYING MENU-DROPS AND HOW TO AVOID THEM

by Dan Moore and David Small

There you are, working on your program. The mouse is near the top of the page. With the delicacy of a microsurgeon, you avoid that overzealous menu bar as you move items around onscreen.

But, oh no! You breathed on the mouse! It jogs up a millimeter and a drop-down menu bops right on top of your file icon—for the fiftieth time this hour! Aiigh!

Tired of this scenario? Frustrated by interfering drop-down menus? Weary of nagging obstacles?

Then what you need is the Dandy Dan Moore MouseTrap!

With this amazing program, you have not one, but *two* programs in one! The first routine, The Berlin Wall, prevents you from ever hitting the menu bar accidentally; it erects a wall between the main screen and the menu bar. Try as you will, you cannot cross up into that menu bar. The second routine, The Amazing Clicker, makes the right mouse button pop you up to the menu bar, no matter where you are. You can be at the bottom of the screen, the top, or (God forbid!) visiting IBM headquarters in Boca Raton, and if you press the right button, boom! You're in the menu bar, and (gasp!) the menus pull down.

I know, you're stunned. But try to be calm. It is all in *one* program, MOUSTRAP.PRG. Slip it inside a sleeping AUTO folder on your boot disk, and you're set.

Just how did I accomplish this miracle? It all started one dark and stormy night. I accidentally pulled down a menu bar, and decided there had to be a better way. Somewhere, a dog was barking. So, I went hunting for the low memory variables that control the mouse. Since they aren't documented in the Atari papers, I had to look for them.

The search method was rather novel. First, using an XBIOS command, I instructed the video chip to begin displaying memory at location 0. This gave me a bit-map view of the first 32K of the hard way.

"Macintize" yaur drap-dawn menus and prevent cursar clutter. The Amazing MouseTrap will lack aut the menu bar until yau really need it. No more accidental drop dawns. Program and listing are in the MOUSTRAP.STQ falder an your START disk. ▸

memory. While moving the mouse around, I looked for blinking pixels which seemed to represent the mouse X and Y registers. Upon finding them, I measured with a ruler from the top of the screen, divided against a 32K screen size, and got an approximate memory location of the blinking pixels. I then used SID, the Developer's Kit debugger, to determine the exact location. No kidding, that's how I did it.

My stunningly commented listing illustrates the action of the program. (See MOUSTRAPS in the MOUSTRAP.STQ folder on your START disk.) Each mouse exception, I intercept a three-byte, relative mouse packet (button, X delta, Y delta), then calculate where the mouse will end up after the current movement is complete. If the mouse will end up at the top of the screen, I cruelly prevent it by zeroing the Y component of the mouse movement. Then, there's a right button check. If the right button is depressed, I tell it to move up 128 pixels four times. This ensures that no matter where it is on the screen, it ends up at the top in one magic jump.

You will note that I wedge this routine in the "mouse-packet exception vector" whose address can be determined with the keyboard-vector XBIOS call.

So much for the heart of the program. How do I make it stay resident in memory? I first tried MOUSTRAP as a desk accessory, but room for those is precious and limited, so instead I went for a terminate-and-stay-resident program designed for an AUTO folder. There is a catch here, alas. After AUTO folder programs are run, GEM reinitializes the mouse vectors through **Initmouse()**, effectively demolishing my own custom vectors. To solve this, I stole the Trap 14 jump vector, and prevented GEM from clobbering my jump table.

Bells and whistles included making it work with the color monitors (less scan lines in the sensitive region), adding real live comments to the code, and polishing off a six-pack from the fridge to get me through.

Users of the few programs that use the right button should note that conflicts *can* occur with the MouseTrap. Don't load it in with DEGAS or NEOCHROME, but otherwise, you will probably be fine. ■

(**Editor's note:** *The MouseTrap code can be adapted to your own nefarious schemes. One of the practical jokers at START created MOUSEGAG.PRG (also in the MOUSTRAP.STQ folder). Sneak it into a friend's AUTO folder and watch the fun.*)

XLEnt Software

Presents

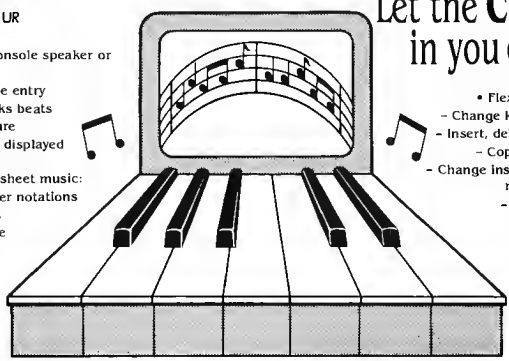
ST MUSIC BOX

copyright 1986

P.O. Box 5228
Springfield, Virginia 22150
(703) 644-8881
Telex 269728 XLNT UR

Let the **Composer** in you come out.

- Compose for ST console speaker or MIDI synthesizer
- Keyboard or mouse entry
- Smart editor checks beats entered per measure
- All parameter info displayed during playback
- Print high quality sheet music:
  - Add lyrics & other notations
  - Include graphics
  - Print single voice or combination
- Conforms to MIDI standard
- Supports monochrome and color



- Flexible composing tool:
  - Change key & time signatures
  - Insert, delete & copy measures
  - Copy from voice to voice
  - Change instruments measure by measure for any voice
  - Alter tempo measure by measure
  - Control synthesizer portamento
  - Capability to transpose notes
  - Load/Save rhythms, single voice or compositions

$49.95

By Dennis Young & Len Dorfman

Includes four compositions!

CIRCLE 130 ON READER SERVICE CARD

Another Great Simulation from Sid Meier – Author of F-15 Strike Eagle

Now he takes you from the cold, thin air and limitless space of F-15 Strike Eagle down into the dark depths of the Pacific Ocean inside an American World War II submarine for a realistic, action-filled simulation —

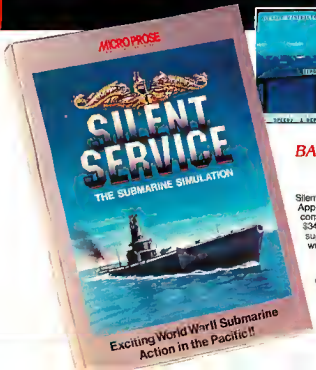
SILENT SERVICE

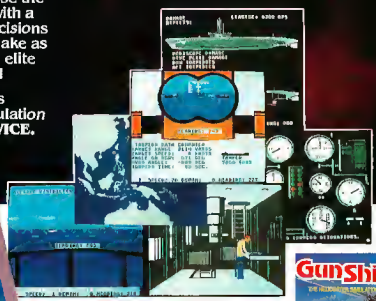
ATARI 520 ST NOW SHIPPING!!

Thrill to the initial sighting of the enemy's strike force in your periscope as their ships come into your range. But watch out — the enemy's escorts have just sighted you. You're the **hunter** — but suddenly — you've become the **hunted!**

As Commander, you must sink their ships and keep your submarine from being destroyed — if you can. Will you select a quiet patrol sector in the Marianas Islands or choose the dangerous waters off the coast of Japan? is a submerged daylight periscope attack best or do you charge in on the surface at night using *only* radar bearings to guide you? Do you fire a spread of your precious torpedoes or can you close the range and pick off the enemy with a single torpedo shot? These decisions and many more are yours to make as you take your place among the elite ranks of the **SILENT SERVICE!**

It's **exciting** — and it's **fun.** It's another great Micro Prose simulation — and it's called **SILENT SERVICE.** Look for it now on your dealer's shelves.





FIVE AUTHENTIC BATTLE STATION SCREENS

Silent Service is available for Commodore 64/128, Apple II Family, Atari XL/XE, IBM PC/PC Jr. computers for a suggested retail of only $34.95. Available soon for Macintosh for a suggested retail of only $39.95. Call or write for more information or MC/VISA orders.

COMMODORE, APPLE, ATARI, IBM and MACINTOSH are registered trademarks of Commodore Business Inc., Apple Computer Inc., Atari Inc., and International Business Machines Inc., respectively

Photo courtesy Baltimore Maritime Museum

Try These Other Real Life Simulations

During Attack Helicopter Action in the AH-64 Apache!

Thrilling Decathlon of aerobatic events in your own personal jet!

You are in command — North Africa 1940-1942

MICRO PROSE
SIMULATION • SOFTWARE

120 LAKEFRONT DRIVE • HUNT VALLEY, MD 21030 • (301) 667-1151

CIRCLE 065 ON READERS SERVICE CARD

THE AI APPRENTICE

EXPLORE EXPERT SYSTEMS WITH XLISP

by Christopher F. Chabris

An introduction to Artificial Intelligence—its languages and its methads—including an examinatian af LISP and a working expert systems pragram written in XLISP. Laak in the falder XADVISOR.STQ an yaur START disk.

Artificial intelligence has at last arrived for the Atari community. With the 16-bit ST line, Atari has provided a computer whose capabilities approach those of advanced systems used for university and industrial research. At a fraction of the price, Atari enthusiasts now have the power to experiment seriously with artificial intelligence in languages like LISP and PROLOG

WHAT IS ARTIFICIAL INTELLIGENCE?

As an idea, artificial intelligence (AI) provokes both fear and philosophical controversy. Many enthusiasts possess an almost religious faith in AI, while others violently resist the very notion that a machine can be made to think like a human being. But though they will dispute its ultimate feasibility, defenders and detractors alike usually agree that AI as a science is mysterious, fascinating, and often misunderstood.

Assuming AI is at least theoretically possible, what exactly do we mean by artificial intelligence? We could discuss this endlessly, but I propose, as a working definition, that AI is a science composed of three closely related areas of investigation:

• ROBOTICS: By this I do not mean the study of industrial robots, mechanical arms, etc., but a general research program whose ultimate goal is the construction by whatever means are necessary of a complete and artificial simulation of a human being. Many AI opponents have criticized this goal of AI▶

JAMES ENDICOTT

because it has serious philosophical and social implications. It is still far from being achieved after 30 years of research.

• COGNITIVE SCIENCE: This is the study of cognition, or the processes, structures, and mechanisms of human thought. Many psychologists working in this area use computer models to help them learn more about the brain.

• APPLICATIONS: Making computers more useful to people by applying the theories and techniques of AI. We will concern ourselves primarily with this engineering side of the discipline, writing experimental programs in AI languages.

Among AI researchers interested in our third objective, there has been considerable confusion over what constitutes an AI program, or application. Again, I adopt a pragmatic definition: AI programs are useful software systems that exhibit some behavior that would be considered intelligent or demanding intelligence if displayed by a human being.

PRACTICAL APPLICATIONS

Applied AI research concentrates on several different areas, most notably the following:

• NATURAL LANGUAGE PROCESSING: Programming computers to understand "natural" languages such as English is important for database management and query. Rudimentary front-end programs for microcomputer databases can translate requests such as "Show me last month's national sales figures broken down by region and product" into the system's query language. Another system under development reads newspapers and composes summaries of their content.

• COMPUTER VISION: Much research is concentrated on vision, an important capability for robots and optical scanners. Very simple programs can be constructed to identify shapes in three-dimensional scenes, compare and match objects, and so on.

• MACHINE LEARNING: The ability of the computer to learn, to acquire new skills and knowledge, may be the "missing link" in AI. Much of the work in learning is exciting and relatively recent, but practical applications have yet to be investigated extensively.

• GAME PLAYING: Chess computers represent a significant achievement of AI research. The best program today is stronger than 99 percent of human players and continues to improve. In backgammon, a computer had already defeated the world champion by the late 1970s.

Atari enthusiasts now have the power to experiment seriously with artificial intelligence.

• SYMBOLIC MATHEMATICS: Machines have always been good at numeric calculations, attaining blazing speeds multiplying, dividing, etc. But in the 1960's AI researchers began to program for symbolic operations, such as differentiation and integration of functions. MACSYMA, a popular expert systems program, was the eventual outgrowth of that early work, and is now used widely by engineers for most of these operations.

• EXPERT SYSTEMS: The most successful and visible outgrowth of AI research, expert systems are programs which mimic the reasoning process of a human expert when attempting to solve a particular class of problems. Expert systems are currently used in a wide variety of applications including law, medicine, electronics, chemistry, geology, finance, and the like. Later, we will explore expert systems in detail and build a simple expert-systems program. But first, we need the tools with which to build it.

AI PROGRAMMING LANGUAGES

Most AI programming is done in specialized languages—not C, Pascal, or assembler (heaven forbid), the languages normally used to develop ST applications—but usually either LISP or PROLOG.

LISP is an acronym for LISt Processing because its primary data structure is the list. It was invented by John McCarthy, the man who coined the phrase "artificial intelligence" in the late 1950's. It is one of the oldest programming languages, and today's main dialect, Common LISP, bears a close resemblance to the early versions. Nevertheless, LISP is still a modern language that is used for at least 90 percent of all AI research programming.

There are several reasons for this popularity. One is that LISP is well-suited for symbol manipulation, as we shall see later. Another is that LISP programs can be both interpreted and compiled. The former allows easy modification and experimentation, desirable in a research setting, while the latter provides the execution speed necessary for a commercial product. Finally, LISP has proved worthy as a general-purpose programming language. Word processors—even entire operating systems have been constructed in LISP, and its use is spreading to other applications.

PROLOG, which stands for PROgramming in LOGic, is a language that was developed in France around 1970 by Alain Colmerauer and his colleagues.

A declarative language, it was a first attempt to realize the ideal of logic programming. Rather than explicitly state which steps the computer is to perform, and in what order they are to be executed (imperative or procedural programming), the programmer declares in logical form the relationships and facts of the algorithms and data, and lets the computer take care of the control. At first this is a difficult structure to grasp, being foreign to most programmers. But PROLOG is quite powerful if used correctly and has been chosen by the Japanese for their vaunted Fifth Generation project.

An important trait shared by LISP and PROLOG is their homogeneity. Programs and data have the same structure, and it is a simple matter for programs to create new procedures or modify themselves during their execution. Try doing that with C or Pascal!

EXPERT SYSTEMS

One of the first scientists to recognize the commercial potential of artificial intelligence was Edward Feigenbaum of Stanford University. He is generally credited with inventing the expert system, a type of AI program that is supposed to be able to solve difficult problems the same way a human expert would: by bringing a large body of explicit knowledge to bear on the situation at hand and somehow using it to deduce a solution.

An expert system normally consists of four parts: a *knowledge base*, the encoding of the knowledge necessary to solve the problems for which the system is designed; the *inference engine*, the software that draws conclusions from the knowledge and the facts of the case; one or more *databases* of relevant facts; and some sort of *user interface*. We could define an expert system as a program that is expert in some area, but then we could consider a calculator desk accessory an expert system in the domain of arithmetic. While this is certainly not untrue (can you multiply faster than your computer?), it trivializes the class of problems for which expert systems are designed: domains without unified theories but rather large bodies of disorganized knowledge that experts must draw upon to make decisions.

Most expert systems in use today are rule-based. Their knowledge is represented as a collection of heuristic rules that specify precisely what conclusions can be made under which circumstances, and sometimes with approximately what certainty of correctness. Such production rules usually take on a relatively simple IF-THEN form like these examples from well-known expert systems:

The ability of the computer to learn may be the "missing link" in AI.

IF: the gram stain of the organism is negative,
the morphology (shape) is rod,
the aerobicity is anaerobic,
THEN: the organism is bacteroides.

IF: the context is layout and assigning a power supply,
an sbi module of any type has been put in a cabinet,
the position it occupies in the cabinet is known,
there is space available in the cabinet for a power supply,
there is an available power supply,
THEN: put the power supply in the cabinet in the available space.

The first example comes from MYCIN, a pioneering medical diagnosis system developed at Stanford over the last fifteen years. The second is from XCON (formerly R1), a program in daily use at the Digital Equipment Corporation to configure VAX computer orders. The two rules share a common structure that looks like this:

IF: <antecedent condition 1>
(AND)
<antecedent condition 2>
(AND)
...
<antecedent condition n>
THEN: <consequent action(s)>
(are all performed)

Normally, each condition can take on a truth value of either TRUE or FALSE. A condition is presumed FALSE until proven TRUE; this is known as the closed-world assumption. The classical rule of inference used to make deductions with these rules is called *Modus Ponens*, and it states the apparently obvious:

> Given a rule: A implies B,
> and given: A (is true),
> we conclude: B (is true).

In our case, this means that if all the antecedent conditions are TRUE, then the consequent actions are performed. These could be simply assignments of truth values to other facts, or maybe entire procedures to be executed (in LISP, expressions to be evaluated).

Note that a rule itself is context-independent; that is, the quantum of knowledge it embodies is always applicable within the domain of the expertise. The XCON rule above uses the first few antecedents as a guard to establish the situations in which it is applicable, but as a whole the rule is context-independent. If we try to use it ▶

in the wrong context, it will just fail with no undesirable side effects, and we can go on to the next one.

Notice that these production rules are a declarative form of knowledge representation, akin to the logic programming concept introduced earlier in the discussion of PROLOG. The rules say absolutely nothing about how or when they will be executed: all the control information comes from the inference engine itself, which decides especially the order in which to use the rules. The rules could have associated "priority" values, for instance, but it would nevertheless be up to the inference engine to refer to those values for guidance.

The program we will examine is XADVISOR.LSP on your START disk. XADVISOR is a production system interpreter (a generic inference engine) written in XLISP that embodies some of the principles introduced above. You may be surprised to see that the entire program, including comments, is just 160 lines long! This includes the inference engine and the user interface, but not the knowledge base. The program will work with any knowledge base in any domain whatsoever as long as its format is syntactically acceptable (as will be described later).

ADVANCED CAPABILITIES

There are several capabilities common in real-world expert systems that are missing from XADVISOR, including the following:

- **CERTAINTY FACTORS:** In diagnosis, we can rarely be completely certain of our results; rather, we have degrees of confidence in the correctness of our reasoning. Some expert systems use statistical tools like "Bayes' Theorem" to calculate the certainty of deductions based on the certainties of the antecedents being true and the certainties of individual rules being applicable. In this

HOW TO USE XADVISOR

The files you'll need can be found on your START disk in the folder labelled XADVISOR.STQ. Copy them to a new disk before using or modifying them. A complete version of XLISP (v. 1.5b) has also been included on your START disk. *Please note:* We have had to compress XLISP in order to fit it on the disk. You must "unsqueeze" the XLISP program before it can be used. See the Disk Instruction page for further instructions on this. Interested readers should obtain the XLISP documentation and source code. There are at least three possible sources:

- **User groups.** Many have a version of XLISP in their public-domain libraries or on their Bulletin Board Systems.
- **CompuServe.** Check Data Library 3 (Utilities) of the ATARI16 Forum. To speed your search, use the keyword "XLISP" (what else?). The version posted here (v. 1.5b) is the same used during the development of XADVISOR and this article.
- **The Catalog.** An XLISP disk has recently been added to the ST Public Domain section.

Wherever you obtain XLISP, be sure to get version 1.5b or higher, as well as the manual and the INIT.LSP file. In the CompuServe distribution, several examples and the complete C source code are available. The following instructions and suggestions assume that you are using XLISP 1.5b; we have not tested other versions at this time, though we have no reason to suspect that later versions (1.6 and up) should behave much differently.

1 If you received an INIT.LSP with your copy of the XLISP interpreter, replace it with the version on the START disk. It ensures that enough memory is allocated for the XADVISOR program and performs several other useful services for you.

2 Make sure that both INIT.LSP and PPRINT.LSP are in the same directory as the XLISP program. Install XLISP as a "TOS-takes parameters" application on the GEM Desktop. If you like, give it a document type of LSP. (It is a good idea to give all your LISP files this extender, for this and other reasons we shall see later.)

3 Locate XLISP and double-click on its icon. A dialog box will open, prompting you for parameters. You can enter as many filenames as you wish, but be sure to leave off the .LSP extender; XLISP will add it to whatever you type. If you don't want to load any files right away, just press [Return].

4 The XLISP interpreter will load and run. First, the message "XLISP version 1.5b, Copyright (c) 1985, by David Betz" will appear at the top of the screen. Second, XLISP will try to open the file INIT.LSP and evaluate all the expressions it contains. Afterwards, or if INIT.LSP cannot be found in the default directory, XLISP will similarly load the files you specified as parameters in the order in which you listed them. Finally, it will enter the top-level READ-EVAL-PRINT loop, issuing a cryptic ">" prompt.

5 At this point, if you did not specify "XADVISOR" as a parameter in the dialog box, type the following.
(load "xadvisor.lsp")
and press [Return]. Note that here, with the **load** function, you must specify the extender in the filename.

6 Once the XADVISOR.LSP file has been loaded successfully, you are ready to consult a knowledge base. Two are supplied on the START disk: HIRING.RUL and CHESS.RUL. For demonstration purposes we'll use the former because it is smaller, but the procedures are the same no matter which knowledge base you use. (I recommend a .RUL extender on all your knowledge bases for mnemonic purposes.)

7 At the next ">" prompt, type
(prepare "hiring.rul")
and press [Return]. This will cause the knowledge base about hiring faculty to be loaded and prepared for consultation. (Note: If you receive an error message from mistyping and end up on a numbered prompt—such as **1:>** – press Control–G until you reach the top-level ">" prompt.)

8 We are ready to ask XADVISOR for advice! Try entering the following for starters (at the prompt):
(advise nil nil 'interactive)
The program will start to ask you questions like this:
Is is true that:
(candidate is one of the top ten in his or her field)?>
Make up some answers (either Yes or No) and see how the program responds. Eventually, you will either receive a suggestion like this:
I advise offering the candidate a senior faculty position.
OK
Or, perhaps this message:
I cannot give any advice based on the given context.
OK
The former represents the best possible advice the program could give based on your answers to the questions. The latter means that the program couldn't draw any meaningful conclusions from your answers.

9 The above method is the easiest way to query XADVISOR, and the only one needed. It is possible, however, to start the program with a list of already known facts, so that you don't waste time answering them, or a "goal" fact—one towards which the system should orient its deduction. The first **nil** in the above command represents the "known facts" parameter, and the second **nil** the goal. A glance at the HIRING.RUL listing will show examples of both statements. Newcomers to LISP will have difficulty establishing the proper sequence of parentheses and quotes necessary to properly input the parameters, but here is an example of how to input a goal parameter: First, **setq** the list to a label, such as "junior":

(setq junior '("advise offering the candidate a junior faculty position"))
Now, use the label within the query:

(advise nil junior 'interactive)

MISCELLANEOUS

The pretty-printing feature implemented in the PPRINT.LSP file is used by typing **(pp function-name)** and pressing [Return], where "function-name" is the name of any function not built in. This is indispensible when debugging your programs—just try to read a function definition rendered by the regular print function. You can also use PPRINT.LSP to pretty-print any list, both within a running program and from the top-level loop.

There is one error in some versions of the XLISP manual that could prove extremely frustrating. In Section 4, "BREAK COMMAND LOOP," it is erroneously stated that invoking the QUIT function within a break loop will return to the next higher level, eventually reaching the top level loop again. *This is not the case.* You must press Control–G to do this. QUIT is not defined and will just get you into the next lower break loop if you try to use it ▶

way, they can offer a range of diagnoses with the approximate probabilities that each is correct, along with prescribed treatments that take this information into account.

- **EXPLANATION FACILITIES:** Good expert systems should be able to not only make deductions, but also to explain how they were made. Whenever the user is asked to provide more data to the program, he should be able to ask why he is being asked the question so that he can follow the system's execution and understand how it arrives at its conclusions. Indeed, many say that an expert system is not truly "expert" if it does not know how it makes its decisions.
- **METALEVEL REASONING:** It is possible to add control strategy to the system's knowledge via "metarules," or rules about the rules. This is becoming increasingly necessary for the needs of large systems for reasons of efficiency. With thousands of rules to choose from, what guarantees an uninformed inference engine will select those rules that are most applicable to the current context (the facts in the database)? Wandering about aimlessly, without outside guidance, it could waste valuable time pursuing dead-end reasoning paths.
- **LEARNING/KNOWLEDGE ACQUISITION:** Expert systems can be costly to maintain and update, so researchers have sought to solve the problem of machine learning within the domain of expert systems. The user should be able to interactively interrogate the knowledge base, and, with the program's assistance, modify its rules to correct bugs or add new information. Randy Davis's TEIRESIAS addition to MYCIN enabled a physician to easily use and update the system unattended.
- **REFINED USER INTERFACE:** XADVISOR's interaction with the user is limited to yes/no questions, whereas the state-of-the-art systems in use today can carry on natural-language dialogs and allow for graphical input of various data. This issue really has as much to do with software engineering as with expert systems themselves.

INFERENCE ENGINE DESIGN

There are two basic methods an inference engine can use to operate on the knowledge and make deductions: goal-oriented *backward chaining* and data-driven *forward chaining*. In the former procedure, the system attempts to establish the truth value of a goal fact with respect to the given data, while in the latter it takes that data and makes all possible deductions until it has proven one, several, or all goal facts.

Backward chaining would be used if we had a goal in mind that we wanted to prove. We would use forward chaining if we had some data but no idea what result it might lead to, or if we wanted to find all possible conclusions reachable from the data. Note that these two control methods will never produce mutually inconsistent results: the set of facts deducible from the data with a set of (context-independent) rules does not depend on the order in which those rules are used. Both strategies are implemented in XADVISOR.

PROGRAM DESCRIPTION

We are at last ready to examine XADVISOR in depth. The XADVISOR sidebar and Disk Instruction page contain startup descriptions. (**Editor's note:** *Thanks to David Betz, author of XLISP, and an effective data compression program from Quack Computer Company, we have found room on the START disk for a complete version of XLISP (v. 1.5b) from which you can run XADVISOR. Interested readers should obtain the complete XLISP package with documentation and source code from the sources listed in the sidebar. The remainder*

AI *PRODUCTS FOR THE* ST

In early June, the following AI products were available or under development for the ST:

XLISP: For the ST, the only LISP actually available at this writing was version 1.5b of XLISP, an excellent and well-regarded public-domain interpreter by David Betz. This, the first ST version, was ported to the ST by J.R. Bammi in 1985. It is written completely in C, and has been implemented on all the popular 16-bit microcomputers as well as under the MS-DOS and UNIX operating systems. XLISP includes extensions for object-oriented programming, but no compiler is currently available. XLISP is available in Data Library 3 of the ATARI16 Forum on CompuServe, as well as from user groups and Antic's public domain library. Be sure to get all the documentation files, and the C source code if you're interested.

CAMBRIDGE LISP: Metacomco is porting their AmigaLISP package to the ST, and it should be available soon through The Catalog. The Cambridge dialect is somewhat different from Common LISP, the emerging standard, but it has a good reputation among those who use it extensively.

Metacamca
5353E Scotts Valley Drive
Scatts Valley, CA 95066
(408) 438-7201
(Expected available thraugh The Catalog; na price set)
CIRCLE 227 ON READER SERVICE CARD

PERSONAL PROLOG: This long-awaited PROLOG environment for the ST from Optimized Systems Software should also be completed by now. According to OSS, it will use a full GEM multiwindow environment, with incremental compilation and interactive program development and testing. PROLOG programs will have access to TOS services, including GEM.

Optimized Systems Software, Inc.
1221B Kentwood Avenue
San Jose, CA 95129
(408) 446-3099
$74.95
CIRCLE 228 ON READER SERVICE CARD

EXPERT OPINION: This first expert systems shell for the ST was developed by MindSoft of France using Modula-2, and is available through The Catalog. It too makes use of the GEM interface. I include it here because it is really just a programming language, or at least an AI tool, that is specialized for producing expert systems. Expert Opinion is as powerful as most of the MS-DOS packages currently available, and is especially suitable for experimenting and prototyping larger systems.

The Catalog
524 Second Street
San Francisco, CA 94107
(800) 443-0100 Ext. 133
$99.95
CIRCLE 229 ON READER SERVICE CARD

of this article is an analysis of the XADVISOR code and assumes some knowledge of LISP. We realize few readers will know this language and recommend they pick up any of the LISP books listed in the Reference section. We have found XLISP to work with most of the example code found in standard LISP books.)

Looking at the XADVISOR code, we find that the listing can be divided into four parts: the top level and initialization functions (ADVISE, PREPARE, IF), the backward chaining module (BACKWARDS, TRY-ALL, BACKWARDS-AUX), the forward chaining module (FORWARDS), and the utility functions used by them (ASSERT, CONDITIONS, ASK-USER, TRUE, TERMINAL). The entire program amounts to these twelve definitions. Before we look at them, let's examine the main global variables used by the inference engine:

RULEBASE is the knowledge base of the system. It is a list of the production rules in the order that they were read from disk. Each rule is represented itself as a list whose CAR is a list of the antecedent conditions and whose CADR is a single consequent action. Individual conditions and actions (facts) are represented as lists, each containing one string literal.

PROVEN and DISPROVEN are the database of the system. They are simply lists that contain facts that have established truth values.

Now, on to the program's functions:

ADVISE is a front-end function used to query the program on a knowledge base that has already been loaded into memory. Its formal parameter list contains the keyword &REST, which binds the next atom to a list containing all of the remaining arguments to the function. Here, we use this to allow the user to pass one or more option descriptors, which are collected in OPTIONS for later use. Currently, only INTERACTIVE is supported, but others could be added without changing the form of the ADVISE call.

Next we have a SETQ with more than two arguments; it simply binds each atom to the succeding value. Initially, we use I-FLAG to indicate whether the interactive option has been selected, we set the list of PROVEN facts to the CONTEXT passed by the user, and we set the list of DISPROVEN facts to be NIL.

The rest of the function is just a COND expression that branches off to either the backward or forward chaining mechanism depending on the arguments to ADVISE. If there was no goal specified but the user did pass a list of known facts, he or she must want to forward chain from those facts. If there was no goal and no context, he or she must want to find the best solution given data that will later be entered interactively. If there was a goal, he or she must want to backward chain from it.

The rest of ADVISE just sets up some nice looking output. PRINC and TERPRI are standard functions that print out objects and issue carriage returns. The only other form here that we haven't seen before is LET. Without going into detail on its syntax, I can say that LET is a powerful construct that sets up local environments, or temporary bindings that only apply within the LET expression and disappear once it is exited.

PREPARE must be called by the user before the first call to advise ADVISE. Given a valid GEMDOS pathname, enclosed in double quotation marks, PREPARE passes it on to LOAD to read in the knowledge base from a disk file. For our purposes, please accept the actions of both PREPARE and IF (another special construct known as a macro) as magic. They are rather kludgey anyhow, and you can probably figure them out with the help of a LISP book.

BACKWARDS is the real meat of the program. This function takes any fact, presumably a consequent action in one ▶

or more rules of the knowledge base, and attempts to prove its truth. It does this in several ways. First, if ACTION is bound to NIL, it calls TRY-ALL (see below). If ACTION has already been proven, we return T. If it has been disproven, we return NIL. Otherwise, we bind CONDITION-SETS to a list containing all the sets of conditions that would satisfy the ACTION. If there are none, meaning there are no rules that mention the ACTION as a consequent action, we try to ask the user whether it is true or not. This is the only place where user interaction occurs, and it only happens if the user explicitly requested it in the call to ADVISE. (If the user felt he had provided all the facts in the CONTEXT variable, he could operate in a sort of "batch mode," just waiting for the system's advice.) On the other hand, if there are conditions on the ACTION, we call BACKWARDS-AUX to deal with them.

TRY-ALL, given a list of RULES-LEFT (to try), will try to backward chain from every "terminal" goal, that is, every goal that is mentioned as a consequent action in one or more rules in the knowledge base, until it succeeds in proving one of them. It will then return that action, but it will return an inconclusive message if it still fails after recursing through the entire RULEBASE list.

BACKWARDS-AUX, given an ACTION and a list of the set's antecedents of rules that mention it as the consequent, will recurse through that list attempting to find a set of conditions that can be satisfied by ASSERT. If a call to ASSERT returns non-NIL, we can add the ACTION to the set of PROVEN facts and return T. Otherwise, if we run out of CONDITION-SETS, we return NIL as we are unable to establish the truth of the ACTION.

FORWARDS is a huge function that forward chains from the facts contained in PROVEN to deduce more facts, eventually either deducing goals or running out of things to deduce. When called from ADVISE, it is initially passed the rulebase reversed (using the built-in function REVERSE) so that the lowest priority rules appear first. It operates by recursing through the rules in this order, attempting to find rules whose consequent actions are not already proven but whose antecedent conditions all are. When this happens, it has deduced a new fact, so it adds that action to the PROVEN list and begins again at the end of the rulebase. When it deduces a goal, it adds it to the list ANSWER-STREAM which will contain all the goals deduced from the given set of PROVEN facts. If it ever runs through the entire rulebase without a single rule firing, then clearly nothing more can be deduced. In this case, if ANSWER-STREAM is non-empty we return it, otherwise we return the inconclusive message.

By now I hope you have noticed that, despite all the parentheses and the convention against comments embedded within functions, LISP programs are relatively easy to read. This is due to LISP's modular structure, long symbolic atom print names, and the self-documenting nature of its function names.

The remaining five functions of XADVISOR offer good examples of recursive list manipulation and I/O. In ASK-USER we print a question about the truth value of a particular ACTION and then call on built-in function READ to obtain the answer. READ is actually the same reader we met early on at the beginning of LISP's top level READ-EVAL-PRINT loop. As shown here, it can be called any time, and will return a symbolic expression obtained from standard input. Of course, EVAL and PRINT are also available to user programs, though they were not directly necessary to this one.

In fact, in my experience, XADVISOR is an excellent example of how much can be accomplished with a very few LISP primitives. We didn't use any fancy object-oriented programming techniques, destructive list modification, anonymous functions, property lists, or other esoteric powerful features of the language. XADVISOR was originally written in Franz LISP, a dialect that runs primarily under the UNIX 4.2BSD operating system on VAX minicomputers. It required changes to only eight lines of the source code to port it over to XLISP running under TOS on the Atari ST.

To help you get started experimenting with XADVISOR and XLISP, I have included two sample knowledge bases on the disk: HIRING.RUL and CHESS.RUL. The former, containing 15 rules, makes a somewhat whimsical attempt to advise a Computer Science department chairman on hiring candidates for faculty positions. The latter, with 57 rules for recommending plans in chess positions, taxes the abilities of a limited production rule interpreter like XADVISOR. With some easy extensions and major additions, XADVISOR could be suited to such a complex problem domain.

CONCLUDING THOUGHTS

I hope this article has raised your interest in artificial intelligence, expert systems, and LISP programming. We had a lot of ground to cover, but you can catch up and learn far more by reading any of the books listed below. With the ST, you have a machine that is definitely capable of serious AI work. One company is marketing the Macintosh as an AI workstation for LISP and expert systems, so why not the faster, higher-resolution ST?

The most difficult tasks for human beings, such as multiplying ten-digit numbers or remembering detailed facts, are simple even for tiny computers. But the simplest second-nature human behavior, such as reading, speech, and vision, is extremely difficult for today's

most powerful computers. These differences--between the machine and the mind--make the challenge of artificial intelligence all the more intriguing! ■

REFERENCE:

For an entertaining introduction to artificial intelligence, very suitable for the non-technical reader:

- *The Cognitive Computer: On Language, Learning, and Artificial Intelligence*, by R.C Schank with P. Childers, Addison-Wesley, Reading, MA

The following are general texts on artificial intelligence. Although they are rather technical, confident programmers should not have much difficulty understanding them:

- *Introduction to Artificial Intelligence*, by E. Charniak and D. McDermott, Addison-Wesley, Reading, MA
- *Artificial Intelligence (second edition)*, by P.H. Winston, Addison-Wesley, Reading, MA

There has been a proliferation of books on expert systems recently. Here are some with which I am familiar:

- *Building Expert Systems*, by F. Hayes-Roth, D.B. Lenat, and D.A. Waterman, Addison-Wesley, Reading, MA
- *Introduction to Expert Systems*, by P. Jackson, Addison-Wesley, Reading, MA
- *A Guide to Expert Systems*, by D. Waterman, Addison-Wesley, Reading, MA

The following all offer excellent coverage of LISP programming, often relating the language directly to AI problems and their solutions:

- *Artificial Intelligence Programming*, by E. Charniak, C. Riesbeck, and D. McDermott, Lawrence Erlbaum Associates, Hillsdale, NJ
- *Common LISP: The Language*, by G.L. Steele, Digital Press, Burlington, MA
- *LISPcraft*, by R. Wilensky, WW. Norton, New York, NY
- *LISP (second edition)*, by P.H. Winston and B.K.P. Horn, Addison-Wesley, Reading, MA

Finally, a few good books on PROLOG and logic programming:

- *Programming in Prolog (second edition)*, by W.F. Clocksin and C.S. Mellish, Springer-Verlag, Berlin, Germany
- *Introduction to Logic Programming*, by C.J. Hogger, Academic Press, London
- *Prolog for Programmers*, by F. Kluzniak and S. Szpakowicz, with J.S. Bien, Academic Press, London, England

Start Your Own BBS! New for the 520 & 1040ST

This user friendly BBS is loaded with special features... Passwords, E-Mail Message Base, X-Modem Protocal col and can run on any number of drives including Hard Drive. Will work with any Hayes-Compatible modem. Complete Package with all Utilities and **Source Code** enabling easy modification and customizing of this BBS.

Everything you need in one package. **$54.95** Plus $3.50 Shipping Add 7% Outside USA

520 ST BACK-UP!

DUPLICATING TECHNOLOGIES IS PROUD TO PRESENT

THE 520 ST DUPLICATOR™

Our Famous 1050 DUPLICATOR has now been converted to service the **ATARI™ 520 ST.** This is a revolutionary, new, software only, disk copy system. Now you can back-up heavily copy-guarded and protected disks. And, most important, if new forms of software protection should appear on the market, Duplicating Technologies will provide you with the necessary software upgrades.

Our exclusive **ST DUPLICATOR™** comes complete with user friendly software and instructions.

Only $39.95 Plus $3.50 Shipping Add 7% Outside U.S.A.

Dealer inquiries are welcome call for quantity price quote

"Constantly Working on New Products and Software."

DUPLICATING TECHNOLOGIES inc.

Formerly Gardner Computing

99 Jericho Tpke., Suite 202, Jericho, N.Y. 11753

TECHNICAL INFO ONLY (516) 333-5504, 5712

WEEKDAY ORDERS ONLY (516) 333-5805

EVE & WKEND ORDERS ONLY (516) 333-5950

CIRCLE 019 ON READER SERVICE CARD

TERMS: We accept American Express, Visa, MasterCard and C.O.D. orders. Foreign orders must be in U.S. dollars. All personal checks allow 14 days to clear. Shipping 7-10 days

SEVEN WORD PROCESSORS EXAMINED

by Ian Chadwick

Which word processor should you buy? START examines seven ST word processors now on the market and looks at future products. An extensive chart compares over 90 common features.

More people use personal computers for word processing than for any other purpose. This is hardly surprising: even rudimentary word-processing software makes typewriters look paleolithic by comparison. Perhaps this marvelous new way of writing is the main benefit you hope to get from your ST computer, but even if it is not, selecting a word-processing program is certain to be one of your early software decisions.

What do you need to know to make an intelligent decision about getting a word processor? It depends on your intended use. If you are a professional writer, the word processor will be a tool far more important than any typewriter. Plan to spend some bucks on it. Carefully consider the requirements your marketplace imposes on your output (including efficiency), and shop for features you will need. If you are a beginner, an inexpensive or free program may be all you'll ever need.

ST software is still in its infancy, and as a result there are no word processors that fully use the potential of the machine. Nonetheless, the word processors now available should meet the needs of many of you, and several new processors will enter the market in the coming year.

FREEWARE

Two free word processors are available that should do the job until you have a better idea of what you really want and need: 1ST Word and ST Writer. Both are from Atari, but are quite different from one another.

1ST Word comes free with the purchase of either the 520 or 1040 ST. A GEM-based program, it uses the mouse on the familiar Desktop, pull-down menus, windows, and any desk accessories the writer specifies. Useful for modest writing projects that don't demand much formatting, it is easy to learn and use for the beginner. Unfortunately, 1ST Word lacks headers, page

numbering, file merge, and common printer options that most users want for any serious or lengthy writing. It is a bit slow for the practiced typist, and has no global reformatting capability, making this a tedious chore in a large document.

ST Writer, on the other hand, is a capable and mature outgrowth of the successful AtariWriter for 8-bit Atari's. It is good enough for moderately demanding work, but lacks the document-oriented features necessary for technical writing and other major undertakings such as scripts and books. It is not GEM-based, however it is free. Atari supplies copies to all dealers and registered Atari users groups. They may legally copy it for you. It also appears on CompuServe as downloadable software in SIG*ATARI, and on Atari's own free BBS (408) 745-5308.

NOW SHOWING

Five commercial word processors are available at this writing: Haba Writer, FinalWord, Let's Write, Regent Word and TextPro. The chart accompanying this article details the features of these five programs, plus the two free ones. You may find one of them suitable for you now.

FinalWord, from Mark of the Unicorn, is the most complete and expensive—$145. Powerful and sophisticated, but not GEM-based, it was designed to emulate Perfect Writer, a very popular program in the IBM environment. Unfortunately, it suffered in the translation, fails to exploit the ST's memory, and has several bugs (see review, "The Final Word," Antic, April, 1986). This is a document processor, i.e., suitable for big projects, and has most of the features a professional writer of complexly formatted material would need.

Let's Write, from Mark Williams Company, sells for $79.95. This tool is a collection of programs that perform the range of tasks required of a word processor, however they are not integrated as state-of-the-art word processors are. Not GEM-based, the main program is MicroEMACS, a

Selecting a word-processing program is certain to be one of your early software decisions.

command-driven text editor derived from the mainframe and academic computer worlds. It is powerful but complex, and definitely not easy to use. Supporting programs format the text, control the printer, check spelling, and communicate online. It doesn't use the mouse, the Desktop, or even help-screens, but once you learn it from its large, indexed manual, it is powerful and reliable.

Regent Word, from Regent Software, is designed for the ST, but is also not GEM-based. The stand-alone word processor costs $49.95, but recently the company combined it with Regent Spell into a package called Regent Pak, that sells for the same price. Be sure to ask your dealer for the Pak when you shop for Regent Word. Regent Word II, 100 percent GEM-based, including built-in spelling checker, was in development at this writing, and may now be available for about $100. In some ways Regent Word is similar to ST writer: command-driven, one file at a time, full RAM available, headers, footers and page numbers, etc. (see *Chart* for comparison), but its command set and output functions are not as full as a professional needs and it lacks some niceties I've learned to appreciate, such as a cursor jump to the beginning or end of a line, multiline headers, and search-and-replace in reverse. These are not fatal flaws, but I'd like to see Regent Word II before buying this package.

Haba Writer, $74.95 from Haba Systems, is a GEM-based program and very easy to learn and use, but make sure you get version 1.D2 or later because earlier versions were buggy. As of August, 1986, Haba Writer will have a spelling checker and mail-merge function. It can keep seven separate files available for work in RAM at the same time and has a WYSIWYG display. WYSIWYG means "What You See Is What You Get," that is, the screen is formatted identically to your printer output. As a part of the GEM world, Haba Writer can enjoy the benefits of RAM-resident desk accessories such as Michtron's Alt, BI's Thunder, or even the Reversi accessory included on your START disk.

TextPro is a German word processor reworked for English and sold in the U.S. by Abacus Software for $49.95. Just available at presstime, information from the company describes it as a "professional quality word processor." It is GEM-based, with optional keyboard commands for menu items. Users can define the function keys—make their own macros—and use TextPro as a text editor for the C language. Eight printer drivers come with it, including one that prints sideways on Epson printers.

COMING SOON

On the horizon are several products expected by Christmas or early spring. Hippo Word (est. $90 from Hippopotamus Software) will support a laser ▸

printer, allowing ST users to enter the desktop publishing arena. Two new GEM word processors—1ST Word Plus (from Atari), and PaperClip Elite (est. $100 from Batteries Included)—will read graphic files into text and print them out within your document. HomeText, from Batteries Included, will permit the writer to reduce complex procedures into a single keystroke (macros) and will have menus that pop-up from the bottom of the screen as well as pull-down from the top menu bar.

The two most talked-about programs under development are Microsoft Write, and Word Perfect. Microsoft Write, an implementation of Microsoft's successful Macintosh program, Word, will be sold under license by Atari for approximately $200. Word Perfect, from Word Perfect, Inc., will be similar to one of the best-selling and arguably the most powerful word-processing programs in the IBM environment. Word Perfect, Inc. says it will cost around $500.

SPELLING CHECKERS

Spelling checkers are becoming essential for any serious writer who uses a word processor. Human proofing is still needed to find errors the checker will miss, e.g. "there" for "their," but even good spellers use these programs as proofreaders, to find typos and other errors.

At this writing there are three stand-alone spelling checkers available for the ST: Hippo Spell (Hippopotamus Software, $39.95), Regent Spell (Regent Software, $49.95), and Thunder (Batteries Included, $39.95). The first two are programs that work on a document after it has been written and saved. Thunder can also work *while* you type. This is a new feature, successful in the IBM PC world and appearing also on expensive electronic typewriters. Time will tell if it is a real advantage or only a gimmicky annoyance.

A basic spelling checker compares each word in your document to the words in its own "dictionary," and alerts you to exceptions. These exceptions may be correct words that you can accept, or errors you can change. Regent Spell and Hippo Spell are basic spellers with 30,000 word dictionaries. Both of these programs allow you to add your own words to the dictionaries.

Regent Word was recently combined with Regent Spell into a package called Regent Pak.

Thunder is more advanced. In real-time mode, it alerts you as you type an unrecognized word, and suggests alternative words you may have meant. It will "learn" your most common errors (e.g. receive for recieve) and automatically correct them when encountered. It has global correction, handy for making a systematic change in a document, and can expand your favorite abbreviations. Thunder also analyzes your document statistically, counting everything from characters to paragraphs, and renders several indexes of readability.

THE CHART

The Word Processor Comparison Chart (on page 36) lists many features that are available on word processors. Each feature can be useful in some situations, but not others; you must decide for yourself, based on your probable use.

CONCLUSION

If you think you need or want a word processor, you have several options available now, or if you want to wait, several important new products are expected within six months. Two major types exist: GEM and nonGEM. If you like working on the Desktop with icons and the mouse, stick with GEM. Non-GEM programs will require learning commands and procedures that can be very complicated but may be worthwhile if you need special powers or features. If you already know a word processor such as EMACS or Perfect Writer, you might shorten your learning curve by getting a lookalike program. Some word processors have their own spelling checker, a value and convenience you should consider. Beginners should start with a program that is easy to use, because complicated and poorly documented word processors can easily become a nightmare. Professional writers need to learn about word processing by experimentation and inquiry in order to evaluate the suitability of programs for themselves; however, they should consider the most capable products available. ■

LIST OF MANUFACTURERS:

- **Abacus Software**
  2201 Kalamazoo S.E.
  P.O. Box 7211
  Grand Rapids, MI 49510
  (616) 241-5510
  CIRCLE 208 ON READER SERVICE CARD

- **Batteries Included**
  30 Mural Street, Unit 9
  Richmond Hill
  Ontario L4B 1B5 Canada
  (416) 881-9941
  CIRCLE 209 ON READER SERVICE CARD

continued on page 36

The Price War Is Over We Won!

We Will Beat Any Price...Anywhere.

Atari ST Hardware

Atari 520ST RGB	$785
Atari 520ST Mono	$645
Atari SF354	$149
Atari SF314	$208
Atari SM124	$165
Atari SC1224	$335
Supra Hard Disk	Call

Printers

Panasonic	Call
Star Micronics	Call
Citizen	Call

Modems

Avatex 1200	$79.00
QMI 1200ST	$129.00
Atari XM301	$35.99
Supra MPP 1000E	Call

Specials

Avatex 1200	
PR Conn. Amodem	$149.00
Atariwriter Plus	$24.95
Avatex 1200	$79.00
Sakata SC100	$137.00
Teknika MJ-10	$178.00
Teknika MJ-22	Call
Habawriter	$25.00
Hippo C	$25.00
Hippo Eprom Burner	$99.99
Color Printer	Call

Avatex 1200 $79.00

Hayes Compatible! Everyday Low Price!

Atari ST Software

Borrowed Time	$29.95
Rogue	$24.95
Final Word	$89.99
Haba Software	Cheap
Hacker	$27.95
H & D Base	$65.00
H & D Forth	$35.00
H & D Toolbox	$25.00
PC Intercomm	$74.95
Personal Diskit	Call
Sundog	$24.95
VIP Professional	Call
Silent Service	$24.00
Flight Simulator	Call
Micro C Shell	$33.00
Mindshadow	$29.95
Print Master	$24.95
P.M. Art Gallery	$19.95
Easy Draw	$97.00
Graphic Artist	Call
Music Studio	$36.00
2Key Accounting	$32.99
Home Planetarium	$19.75
Joust	$19.75
CP/M Emulator	$32.99
Temple of Apshai Trilogy	$24.95
Metacomco Pascal	$69.95
Lattice C	$98.00
Andro	Call
UCSD Pascal	Call
Regent Spell	Call
Regent Word	$35.00
Regent Word II	Call
Regent Base	Call
Meta 68000 Asm	$59.95
Brattacas	$29.95
Degas	$23.95
Personal Pascal	$48.00
ST Talk	$11.00
Zoomracks	$48.99
Modufa-2	$49.95
Personal Prolog	Call
Kings Quest II	$32.00
Tos Chips	$24.95
Universe II	$49.00
ST Copy	$18.99
Typing Tutor	$16.75
Spiderman	$12.95
Mouse Mat	$7.49
Ultima II	$37.50

Atari 8-Bit Hardware

Atari 130XE	$118.95
Atari 65XE	Call
Atari 1050	$124.50
Indus GT	Call
Atari 1027	$79.00

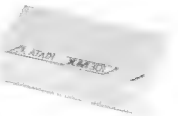
Accessories

US Doubler	$48.00
Rambo XL	$33.00
R-Time Cart	$48.00
Bonus SS DD 5.25	$6.50
Bonus DS DD 5.25	$7.00
Paper 1000 Sheets	$11.00
Paper 2500 Sheets	$22.00
Maxell 3.5	$17.50
Xetec	$39.95
Logo Manuals	$13.50

Interfaces

P.R. Connection	$57.00
Supra Microprint	$29.00
Supra 1150	$39.00
ST Modem Cable	$12.00
Supra Microstuffer	$59.00
U-Print	$48.00
Apetace	$35.00
Atari 850	$109.00
ST Printer Cable	$12.00

Atari XM301 Modem $35.99



Atari 8-Bit Software

Action!	$46.00
Basic XE	$46.00
Basic XL	$36.00
Battalion Command	$24.95
Battle of Antietam	$32.00
B-Graph	$24.95
Broadsides	$24.95
Computer Baseball	$24.95
Computer QB	$24.95
Conflict in Nam	$24.95
Crusade in Europe	$24.95
Silent Butler	$19.99
Gemstone Warrior	$22.00
War in Russia	$48.00
Dec. in the Desert	$24.95
Kennedy Approach	$19.50
Solo Flight	$19.50
Learning Phone	Call
Syncalc 130XE	$29.95
Paperclip	$34.00
Mac 65	$46.00
Print Shop	$27.50
Karateka	$18.50
Lode Runner	$22.00
Atariwriter Plus	$24.95
Panzer Grenadier	$34.00
Hacker	$15.75
Space Shuttle	$15.75
Mindshadow	$15.75
Great Am. Rd. Race	$15.75
F-15 Strike Eagle	$19.95
Silent Service	$19.50
Synfile 130XE	$29.95
Page Designer	$18.00
HomePak	$30.00
OSS Toolkits	$19.50
Printshop Lib. 1/2/3	$17.00
Chmp. Lode Run	$18.50
Ultima IV	$39.00
Sparta Dos CS	$26.00
Home File Manager	$7.99
Music Painter	$7.99
Star Raiders	$7.99

Black Patch Systems

ORDERS ONLY: Call TOLL FREE 1-800-ATARI-02 or 301-987-2300 (Toll Call)

For technical information or order inquiries, call 301-987-0019, or write
Black Patch Systems, P.O. Box 501, Arnold, MD 21012

HOW TO ORDER: CASHIER CHECK, MONEY ORDER, MASTERCARD® OR VISA® (ADD 4% FOR CHARGE CARDS)... NO PERSONAL CHECKS NO C.O.D.s. SHIPPED U.P.S. ALL PRICES SUBJECT TO CHANGE WITHOUT NOTICE
SHIPPING: ADD $3.00 ON ALL ORDERS UNDER $100.00 ADD $5.00 ON ALL ORDERS OVER $100.00. ACTUAL FREIGHT CHARGED ON MULTIPLE ORDERS.
INTERNATIONAL: ACTUAL FREIGHT CHARGED ON ALL ORDERS OUTSIDE THE CONTINENTAL UNITED STATES INCLUDING A.P.O.
POLICIES: NO RETURNS WITHOUT A RETURN AUTHORIZATION NO RETURNS UNLESS DEFECTIVE. ALL DEFECTIVES WILL BE EXCHANGED. NO EXCEPTIONS

CALL OR WRITE FOR FREE CATALOG · DEALER INQUIRIES INVITED

CIRCLE 009 ON READER SERVICE CARD

WORD PROCESSORS...

- **CompuServe Information Services**
  P.O. Box 20212
  5000 Arlington Center Blvd.
  Columbus, OH 43220
  (800) 848-8199
  CIRCLE 210 ON READER SERVICE CARD

- **Haba Systems**
  6711 Valjean Ave.
  Van Nuys, CA 91406
  (818) 989-5822
  CIRCLE 211 ON READER SERVICE CARD

- **Hippopotamus Software**
  985 University Avenue, Suite 12
  Los Gatos, CA 95030
  (408) 395-3190
  CIRCLE 212 ON READER SERVICE CARD

- **Mark of the Unicorn**
  222 Third Street
  Cambridge, MA 02142
  (617) 576-2760
  CIRCLE 213 ON READER SERVICE CARD

- **Mark Williams Company**
  1430 W. Wrightwood Ave.
  Chicago, IL 60614
  (312) 472-6659
  CIRCLE 214 ON READER SERVICE CARD

- **Microsoft Corp.**
  16011 NE 36th Way
  Box 97017
  Redmond, WA 98073-9717
  (206) 882-8080
  CIRCLE 215 ON READER SERVICE CARD

- **Regent Software**
  7131 Owensmouth–Suite 45A
  Canoga Park, CA 91303
  (818) 882-2800
  CIRCLE 216 ON READER SERVICE CARD

- **Word Perfect Corp.**
  (was Satellite Software, Inc.)
  288 West Center
  Orem, Utah, 84057
  (801) 227-4299
  CIRCLE 217 ON READER SERVICE CARD

START CHART

ST WORD PROCESSORS

PROGRAM:	ST Writer	IST Word	Haba Writer	Final Word	Regent Word	Let's Write	Text Pro
Original system	800	ST	ST	IBM	ST	Mainfr	ST
Feature:							
GEM based	no	yes	yes	no	no	no	yes
Protected disk	no	no	yes	yes	yes	no	yes
Handles folders well	yes	yes	yes	no	no	no	yes
Handles hard drive well	yes	yes	yes	no	no	yes	yes
Uses all free RAM for text	yes	yes	yes	no	yes	yes	yes
Text file size limitations	RAM	RAM	RAM	48K	RAM	RAM	RAM
Number of files in memory	1	4	7	10	1	8	1
Help screens	no	yes	no	yes	yes	no	no
Search for bare carriage return	yes	no	no	yes	no	no	yes
Saves ASCII file	no	yes	yes	yes*	yes	yes	yes
Saves unformatted ASCII file	no	no	yes	yes	no	yes	yes
Auto section numbering	yes	no	no	yes	no	yes	no
Auto indexing	no	no	no	yes	no	no	yes
Transpose characters	no	no	yes	yes	no	yes	no
Transpose words	no	no	no	yes	no	no	no
Undo last delete (not cut)	yes	no	no	yes	yes	yes	no
Auto header/footer	yes	no	no	yes	yes	yes	yes
Vary headers/footers each page	yes	no	no	yes	no	yes	yes
Multiline auto header/footer	2	no	no	yes	no	yes	yes
Format disk	yes	no	no	no	no	no	no
Merge files from disk	yes	no	no	yes	yes	yes	yes
Delete file from disk	yes	yes	no	no	no	no	yes
Page wait command	yes	no	no	yes	yes	yes	yes

PROGRAM:	ST Writer	1ST Word	Haba Writer	Final Word	Regent Word	Let's Write	Text Pro
Set starting page number	yes	no	no	yes	yes	yes	yes
Hide format symbols	no	n/a	yes	no	no	no	yes
Auto footnoting	no	no	no	yes	no	yes	no
Uses mouse	no	yes	yes	no	no	no	yes
Keyboard commands	yes	no	no	yes	yes	yes	yes
Uses function keys	yes	yes	yes	yes	yes	no	yes
User-definable function keys	no	no	no	no	no	no	yes
Display 80 cols on screen	yes	no	no	yes	yes	yes	no
Edit > 80 columns/screen	no	yes	yes	no	no	no	yes
Edit > 24 lines/ monochrome	yes	no	no	no	no	yes	no
Change display color	some	yes	yes	no	no	no	yes
Variable line spacing	yes	yes	no	yes	yes	yes	yes
Variable paragraph formats	yes	yes	yes	yes	yes	yes	yes
Auto indentation/variable	yes	no	yes	yes	no	yes	yes
Upper/lowercase toggle	yes	no	no	yes	no	yes	no
Double column editing	no	no	no	no	no	no	no
Math functions internal	no	no	no	no	no	no	no
Save custom print formats	yes	no	yes	yes	no	yes	yes
Multiple fonts or typefaces	no	yes	yes	no	no	no	yes
Word wrap on/off toggle	no	yes	no	yes	no	no	yes
Switch disks while editing	yes	yes	some	no	yes	yes	yes
Sub/superscript commands	yes	yes	yes	yes	yes	no	yes
What you see is what you get	no	yes	yes	no	no	no	no
Auto reformat existing text	yes	no	no	yes	yes	yes	yes
Global reformat	yes	no	yes	no	yes	yes	yes
On screen justification	no	yes	yes	no	no	no	no
On screen text alignment	no	yes	yes	no	no	no	no
Insert/replace mode toggle	no	yes	yes	yes	no	no	yes
Hyphenation	no	no	yes	no	no	no	es
Page/screen up & down	yes	yes	yes	yes	yes	yes	yes
Cursor to line end/start	yes	no	yes	yes	no	yes	yes
Display full character set	no	yes	no	no	no	no	no
Display international character set	no	yes	yes	no	no	no	yes
Display control characters	yes	yes	no	yes	yes	no	yes
Search/replace non-ASCII character set	no	no	yes	no	no	no	yes
Word count	no	no	no	no	yes	yes	no
Character count	no	no	no	es	no	no	yes
Line count	yes	no	yes	yes	no	yes	yes
Available RAM count	yes	no	no	no	yes	yes	yes
Search/replace in reverse	yes	yes	yes	yes	no	yes	no
Ignore case in search	no	yes	no	yes	no	yes	no
Autosave	no	no	no	yes	no	yes	no
Shows page breaks on screen	no	yes	yes	no	no	no	yes
Shows current page number	no	some	yes	no	no	no	yes

START CHART...

PROGRAM:	ST Writer	1ST Word	Haba Writer	Final Word	Regent Word	Let's Write	Text Pro
Can read other word-processor files	yes	yes	some	yes	some	ASCII	yes
Can read ASCII non word-processor files	yes	yes	yes	yes	no	yes	yes
Include graphics with text	no	no	no	no	no	no	no
Macros	no	no	no	no	no	yes	yes
Set multiple marks in text	no	yes	no	yes	no	no	no
Size of manual	med	med	small	large	small	large	med
Manual indexed	no	no	yes	yes	no	yes	yes
Manual on disk or printed	disk	disk	pntd	pntd	pntd	pntd	pntd
Tutorial included	yes	yes	no	yes	no	yes	yes
Print preview	yes	no	no	no	yes	no	yes
Printer > 80 columns	yes	yes	no	yes	no	yes	yes
Print disk directory	yes	no	no	no	no	no	no
Send special printer codes	yes	no	yes	no	yes	yes	yes
Print selected pages	no	no	yes	yes	no	yes	yes
Print from any page	yes	no	yes	yes	no	yes	yes
Double column printing	yes	no	no	no	no	yes	yes
Variable paragraph spacing	yes	no	no	yes	no	yes	no
User configure printer driver	yes	no	yes	yes	ltd	ltd	yes
Multiple printers drivers	no	yes	2	yes	no	no	6
Separate print program	no	yes	no	yes	no	yes	yes
Printing pre-format required	no	no	no	yes	no	yes	no
Draft print option	no	no	no	yes	no	yes	no
Proportional print support	yes	yes	no	yes	no	no	no
Print formatted file to disk	yes	no	no	yes	no	yes	yes
Include or chain external files at print time	yes	no	no	yes	no	yes	yes

The following rated 1-5 where 5=highest value:

	ST Writer	1ST Word	Haba Writer	Final Word	Regent Word	Let's Write
Ease of learning	3	5	5	2	3	1
Complexity of documentation	3	2	1	5	2	5
Completeness of documentation	5	5	4	3	3	5
Clarity of manuals	4	4	5	2	3	4
Suitable for beginner	3	4	5	1	3	1
Suitable for professional	4	2	3	5	1	3
Ease of use	3	4	5	1	3	1
Price range (5=high)	0	0	2	5	2	4
Use of available memory	5	3	4	2	4	5
Use of disk/drives	5	5	4	3	2	3
Quality of screen display	5	5	5	5	4	5
Display speed	5	3	3	5	5	4
Print flexibility	3	4	3	5	2	5

(**Editor's Note**: *Data items for Let's Write provided by Sol Guber. Data for TextPro provided by Arnie Lee, President of Abacus Software. Other data by author. Because items in last chart are subjective, TextPro is not included.*

SWAPPING ART WITH OTHER COMPUTERS

IFF STANDARDS FOR THE ST

by Tom Hudson

The Interchange File Format (IFF), originally developed for the Amiga computer, is now available on the ST. Graphics block transfers—true "clip art" — and Amiga picture compatibility are provided, courtesy of Tom Hudson. The author of DEGAS Elite and CAD-3D explains how to use this new graphics standard. Folder IFF.STQ on your START disk contains all necessary source code for the routines, two sample programs to read and write your own IFF picture blocks, and some sample picture segments.

SWAPPING...

While developing DEGAS Elite, the full-featured version of DEGAS, I decided the program should have true "clip art" capability, such as that found on the Macintosh. This ability to transfer portions of a graphics display (bit-blocks) from one program to another has been missing from the Atari ST. It's not that it hasn't been done, it's just that ST developers have not agreed upon a standard file format.

Originally, I created a very simple bit-block format for DEGAS Elite pictures that would work fine for Atari users. But what if someone wanted to import images from, say, the Commodore Amiga computer? A simple Atari-only format just wouldn't do.

I consulted several prominent ST developers for their opinions, and we decided that Atari bit-block images should be stored to disk in the Electronic Arts Interchange File Format (IFF).

The Interchange File Format is a set of file standards developed by Electronic Arts so that computers of all types can easily exchange a variety of files: graphics, music, text, and so forth. Originally implemented on the Amiga, this format will be the perfect way for the ST to communicate with any other computer that supports IFF.

Why support IFF files? Take a look at the computer industry. Every computer manufacturer has a different notion of how things should be done. You can't use an Amiga or Macintosh disk in an ST drive—they use different formats to store their data. You can't plug an IBM graphics card into an Apple II. The major drawing programs on the ST alone use different picture storage formats! The computer industry needs standards, and the IFF file is a move in the right direction. With it, we take an important step toward computer compatibility.

With a proper standard defined and adhered to by the many ST software developers, we will have a uniform method for exchanging graphics information between applications such as paint programs, page-layout programs, charting programs, or word processors. One program can create a graphic image and write it to a disk file, then another program can use that image in whatever way is required. In addition, it will be much easier to port software from one computer to another because they will use the same file-handling routines!

This article will describe a uniform method of storing graphic bit-blocks in IFF disk files, and provide commented C source listings demonstrating the reading and writing of the files. In addition, the START disk contains example block image files along with the ready-to-use source code, so that START readers can begin supporting the file format immediately.

(Editor's note: *START agrees with Tom Hudson and the developers supporting him on this. We urge all ST developers to adopt the IFF format as an ST graphics standard. Nonprogrammers and others who just want to use the IFF programs on their START disk will find instructions near the end of this article. We advise reading the article for information that will make the instructions clearer.*)

MEMORY FORMS

Before examining IFF and adapting it to the ST, we must understand how the ST itself transfers bit blocks. Whenever the ST's graphics routines perform a bit-block transfer, they must be informed how the graphic information is laid out in the system memory. This is known as a *memory form*, and may be in several formats depending upon the image's resolution and how the program created the image.

Typically, the ST programmer uses the GEM VDI bit-block manipulation functions, **vro_cpyfm( )** and **vrt_cpyfm( )**, to move image blocks around. These functions are passed the format of the bit-mapped image with an array of values known as a *Memory Form Definition Block* (MFDB). Most applications will require the use of the **vro_cpyfm( )** function (opaque color form to color form operation) rather than the **vrt_cpyfm( )** function (transparent, single-plane form to color form) because the latter is mainly used to turn a monochrome image (such as a block of text) into a color image. We will only work with the **vro_cpyfm( )** function in this article.

An MFDB is a ten-word array of values in which each word entry is a part of the information defining the bit-block. Its format is as follows:

MFDB[0]—High word of memory form starting address
MFDB[1]—Low word of memory form starting address
MFDB[2]—Form width in pixels
MFDB[3]—Form height in pixels
MFDB[4]—Form width in words
MFDB[5]—Form format flag
MFDB[6]—Number of memory planes
MFDB[7]—Reserved (0)
MFDB[8]—Reserved (0)
MFDB[9]—Reserved (0)

The first two entries of the MFDB array are the longword address of the start of the bit-block broken into high and low words. These two word values tell the VDI where the bit-block memory form is located in memory. For memory forms that are standard ST screens, this value will be the base address of the screen RAM.

The third entry in the MFDB array is the form width in pixels. This is quickly calculated as:

Rightmost pixel number−Leftmost pixel number+1

For an entire ST screen, this value will be either 320 (low-resolution mode) or 640 (medium- and high-resolution modes).

The fourth entry in the MFDB array, the form height in pixels, is equally easy to calculate:

Bottommost pixel number−Topmost pixel number+1

For an entire ST screen, the height in pixels will be either 200 (low- and medium-resolution modes) or 400 (high-resolution mode).

The fifth entry tells the VDI in which format the bit-block image is stored. There are two formats supported by the VDI: standard form (1) and device-specific form (0). For our purposes, we will use only the device-specific form, since this is the format used in ST screen image data.

What if someone wanted to import images from, say, the Commodore Amiga?

The sixth MFDB value, the form width in words, is the number of 16-bit words required to hold one line of pixels. In C, this value can be calculated as:

(Width in pixels+15)>>4

Finally, the number of memory planes is either 4 (low-resolution), 2 (medium-resolution) or 1 (high-resolution monochrome).

The last three entries in the MFDB array are reserved for future use in the VDI. Set them to zero, to avoid unexpected results in future versions of the VDI.

Once you have properly defined the MFDBs for the source form (where the bit-block is being moved from) and destination form (where the bit-block is being moved to), you can use **vro_cpyfm()** to move a bit-block from one memory form to another. You can use the same MFDB for both the source and the destination if needed, but it's best to perform a bit-block transfer (bitblt) using two separate memory forms to avoid overlap conflicts.

MOVING A BIT BLOCK

Moving a block around is actually very simple, but you must know where the block is and where you want it to go. To do this, you must set up two MFDB arrays, one for the source form and one for the destination form. Normally, one of the memory forms will be a screen RAM area, and the other will be a user-defined holding buffer for the bit-block.

Setting up a user-defined portion of RAM for the image buffer is quite easy. Calculate the size of the picture in bytes.

Number of planes*Width in words*2*Height in pixels

This value is then used as a parameter for the memory allocation function, **Malloc()**. Note that it isn't necessary to allocate an entire 32000 bytes (the full size of an ST screen) for your bit image. For a low-resolution (4 bit planes) bit-block 33 pixels wide (3 words) and 12 pixels high, the RAM required is only (4*3*2*12), or 288 bytes. Calculate the RAM needed, allocate it, and plug the starting address of the allocated RAM into the first two entries of the MFDB. (We'll look at an example of this later.)

Once you have both MFDBs defined with properly allocated sections of RAM, you need to set up another array which tells **vro_cpyfm()** which part of the source form you want to move, as well as where to place it in the destination memory form. This is an eight-entry array with the first four words defining the source rectangle, and the last four words defining the destination rectangle. For example, to move the rectangle with upper-left coordinates 30,40 and lower-right coordinates 50,45 to a rectangle on the destination form with upper-left coordinates 0,0 and lower-right coordinates 20,5 (be sure the two rectangles are the same size), the coordinate array (we'll call it **COORD**) will look like this:

```
COORD[0]=30
COORD[1]=40
COORD[2]=50
COORD[3]=45

COORD[4]= 0
COORD[5]= 0
COORD[6]=20
COORD[7]= 5
```

Now all the preliminary data is set up. We have properly described the memory forms and the portion of the source form we want to move. We're ready to do the bitblt operation with the **vro_cpyfm()** call. It is structured as follows:

vro_cpyfm(handle, wr_mode, COORD array, source MFDB, destination MFDB)

The **handle** is the device handle of the graphics workstation you are using, assigned when you open the virtual workstation with the **v_opnvwk()** function.

The **wr_mode** is the writing mode to use in the bitblt operation, numbered from 0-15, and described in the GEM VDI manual, page 6-6. For our purposes, we will always use mode 3 (replace mode), where the source block is copied into the destination form exactly as it appears in the source form. The other 15 writing modes can be used for special masking effects, and you can experiment with them to see what they do.

This format will be the perfect way for the ST to communicate with any other computer that supports IFF.

The **COORD** array is the eight-word array described earlier which tells the VDI the coordinates of the bit-block to move.

The **source MFDB** is the pointer to the MFDB of the form we're moving the block from.

The **destination MFDB** is the pointer to the MFDB of the form we're moving the block to.

When **vro_cpyfm()** is called, the bit-block will be copied from the source form to the destination form, and if the two MFDBs are different formats (as in the source form being a standard ST screen and the destination form being just the size of the bit-block), the VDI will convert the form's format accordingly.

INTERCHANGE FILE FORMAT

Now that you know the basics of setting up a memory form and copying a bit-block from one form to another, let's see how to store bit-blocks in a disk file for later use. This is where the IFF file format comes in.

The IFF routines on your START disk are a subset of the full IFF specification. These routines can handle bit-mapped images of various resolutions quite well but do not handle text or any of the other IFF file types. They are easy for the ST programmer to use because they take standard GEM VDI Memory Form Definition Blocks and ST palettes as input; the programmer doesn't have to deal directly with the IFF files. When reading the IFF file, the routines take care of most conditions that can occur, and return palette information in both a system palette table and a GEM color format. In short, the routines make reading and writing IFF files a quick and painless task for the ST programmer.

The IFF bit-block files we'll be using have a standard format, shown below. For ST IFF bit-block files, I suggest three standard extenders: For low-resolution (16-color) images, use .BL1; for medium-resolution (4-color) images, use .BL2; and for high-resolution (monochrome) images, use .BL3. These correspond to the .PI1/.PI2/.PI3 extenders used by the DEGAS paint program, which allow users to quickly identify the resolution of the image contained in the file. This will become important when images begin to appear on bulletin board systems with little documentation.

Each IFF bit-block file begins with the four-character identifier FORM followed by a four-byte longword indicating the length of the file (*not* counting the FORM header and length code). The program reading the IFF file should check for this header to be sure the file is in the proper IFF format.

Following the FORM header is the four-character code ILBM, which stands for InterLeaved Bit Map (not to be confused with ST Bit-Plane Interleaving). This code lets the reading program know the file contains an IFF bit-block image.

Next comes another four-character identifier, BMHD, which stands for Bit Map HeaDer. It is followed by a four-byte longword indicating the length of the header. The length is expected to be 20 bytes with the following structure:

Byte Offset	
0	word—Width of form in pixels
2	word—Height of form in pixels

4 word—X pixel location (assumed to be 0)
6 word—Y pixel position (assumed to be 0)
8 byte—Number of bit planes
9 byte—Masking flag—unused in our routines
10 byte—Compression flag—0=No compression, 1=RLE
11 byte—Unused filler byte
12 word—Transparent color pixel value (normally 0)
14 byte—X Aspect ratio (unused by our routines)
15 byte—Y Aspect ratio (unused by our routines)
16 word—Page width
18 word—Page height

Following the BMHD data chunk is the color palette information. This is another four-character identifier, CMAP (Color MAP), and it is also followed by a four-byte longword indicating the number of bytes in this data chunk. The data in this chunk is a group of three-byte structures, as shown below. Each three-byte RGB value represents one color:

byte—Red value for color
byte—Green value for color
byte—Blue value for color

The color information in each of these bytes must be aligned with the most significant bit. For example, the ST's palette registers contain red, green and blue settings ranging from 0-7, which requires three bits to store. In a byte, these bits are stored like so:

00000XXX

To make the value conform to the IFF format, we shift the color-register value to the left five bits, so that the most significant bit of the byte contains the most significant bit of the color data, like so:

XXX00000

For a low-resolution image, the color map contains 16 three-byte entries; for a medium-resolution image, it contains four three-byte entries; for a high-resolution image, it contains two three-byte entries.

One problem arises at this point. When transporting bit-block files from the Amiga (which allows up to 32 colors) to the ST (which allows only 16 colors), the ST routine must drop some of the colors and their corresponding bit planes. When this happens, the first 16 colors are used, and the rest are discarded. The same applies for the extra bit-planes. The IFF routines supplied here can read one resolution bit-block and turn it into another resolution, but be warned that reading a bit-block image file into a memory form with fewer bit planes (such as reading a .BL1 file into monochrome mode) will probably render the image useless. You *can* read blocks with fewer bit planes into a form with more bit planes with no trouble—the colors will show up properly.

After the color map, the last data chunk in the IFF file is the bit-mapped image itself. This is preceded by a four-character identifier BODY and a four-byte longword specifying the length of the bit-block data.

The BODY can be written in two different forms: uncompressed and compressed. The compressed format uses a simple Run-Length Encoding RLE scheme (see *Reference 2*). Note that the IFF Reader on your START disk will read both uncompressed and RLE-format bit-block data, but, for simplicity's sake, the write routines will only write uncompressed data. If you like, you can modify the write routine so that it supports the RLE option.

This is a simple overview of the IFF bit-block format. As I said earlier, the IFF routines do most of the work for the programmer and a detailed knowledge of IFF format is not needed to use them. However, if you would like to get full information on compression format and other details, you should read the Electronic Arts "EA IFF 85 Standard for Interchange Format Files" document (see *Reference 1*).

IFF FILE HANDLING ROUTINES

The file IFFRTNS.C on your START disk is the C source code for the actual IFF I/O routines. To use this in your programs, compile it into a .O (object) file and place it on your linker disk. Whenever you want to use the IFF routines, link the IFFRTNS.O file with your program, and you're ready to go.

There are three routines in the IFFRTNS.C file that you will be concerned with: **iff_rd1( )**, **iff_rd2( )** and **iff_wrt( )**. All these routines and their support functions are documented for your use.

The IFF I/O routines make several items available. The **Syspal[]** array is a 16-word array which holds the color-palette information as used by the **Setpallete( )** call. That is, each entry ranges from $0000-0777, where $0000 is black and $0777 is white.

The **Gempal[16][3]** array is an array of color values used by the GEM **vs_color( )** function. This array has 16 groups of three values, where **Gempal[][0]** is a value from 0-1000 for the red portion of the color, **Gempal[][1]** is the green component, and **Gempal[][2]** is the blue component.

Finally, the **xparent** variable is an INT value that tells which color is assumed to be transparent. This will typically be the background color index, 0.

The colors in a block file may differ from the color palette ▶

that your program is using. Because of this, the block images may not look correct when loaded. With the color-palette information supplied with each block, you can do what is termed a "color re-map" – that is, you can process the pixels in the block image and replace the old pixels with the color from the current palette which most closely matches the original pixel color. This rather complex operation requires specialized assembly language and is beyond the scope of this article.

IFF I/O FUNCTIONS

The **iff_id()** function, used internally in the IFF I/O routines, reads a four-character data ID and the four-byte length into the **chk** structure.

The **proc_CMAP()** function is used after the color map (CMAP) chunk has been located. It reads the color values from the block file and transforms them into both the ST BIOS format (in the **Syspal**[] array) and the GEM format (in the **Gempal**[][] array).

The **decompress()** function takes care of reading bit-block images which were written using the RLE compression scheme. Decompression is done one scan line at a time.

The **rawread()** function does the same thing as **decompress()**, but with block image data that is not compressed.

The **rawwrite()** function writes a scan line of uncompressed data to a disk file. As mentioned, there is no corresponding **compress()** function here, but you can add one to write image data in a compressed form, if you like. The Read routines can handle either format. Compression on writing was not supported primarily, because small block files benefit little from RLE compression.

The **toraster()** function does the job of converting the raw, uncompressed scan line data into the ST's device-specific "interleaved" bit-map format. This screen data format is discussed in the article "GRAFCON ST" in the July, 1986 issue of Antic magazine.

The **fmraster()** function does the opposite of the **toraster()** function. That is, it takes a scan line of device-specific data and converts it into the form required by the IFF data file. After this routine, the data is ready for writing (or compression, if you choose to install RLE compression).

The **iff_rd1()** and **iff_rd2()** functions are the actual functions called to read an IFF file. This is a two-phase operation: You furnish an MFDB array and the handle of a file opened for input, and call **iff_rd1()**. This function reads the starting portion of an IFF file, sets up the MFDB array to let you know how the block is laid out, and returns. Using the information in the returned MFDB, you allocate the amount of RAM the block needs and call **iff_rd2()** to complete the read process. This operation is shown in detail below.

The **iff_wrt()** function is used to write a bit-block to disk. You furnish the handle of a file opened for output, the MFDB of the block, a color palette in the ST BIOS format, and the resolution of the image. This process is also shown in detail below.

READING IFF FILES

As mentioned, the IFF file-read process is a two-stage process. IFFREAD.PRG, on your START disk, is a program demonstrating the reading of IFF block files. Sample block files of the three different resolutions are supplied on the START disk. Files with .BL1 extensions are for low-resolution, .BL2 files are for medium-resolution, and .BL3 files are for monochrome. Try reading various resolution block files in the ST's three graphics modes to see how each looks. The following is an explanation of the source code found in IFFREAD.C.

IFFREAD starts by opening a virtual workstation and requesting the screen's location in memory and its resolution. The color palette is also placed in the **oldpal**[] array for later restoration. Important: *always* restore the color palette after your program is done so that when the system returns to the Desktop, it is the same as it was before your program executed. This is not only considerate to the user, but it insures that the Desktop screen will be readable!

Next, the program builds the screen's MFDB array. This is a fairly straightforward operation. As you can see, it plugs the appropriate values into all ten **scrnmfdb**[] array locations. Be sure to always place zeroes into the last three MFDB locations to insure compatibility with future GEM VDI releases.

Now we bring up the GEM file-selector dialog and let the user select the IFF block file to load. Once the filename is entered, the program displays the **whichfm** dialog to ask the user where to load the block image.

The **whichfm** dialog lets you choose how the block file will be loaded. If SCREEN is chosen, the block will be loaded into the screen's MFDB, regardless of the block's size. In this case, the block is read into the upper-left corner of the screen, showing that a block can be read into any size memory form. Of course, the only usable portion of the form is the block itself.

If FORM is chosen in the dialog, the program will allocate the amount of RAM required for the bit-block in a part of RAM separate from the screen, read the block into that RAM, then do a bitblt to copy the block to the center of the screen. This is a triple-edged demonstration, showing how to allocate RAM, bring in a bit-block to a special buffer, and how a bitblt operation works.

After SCREEN or FORM is chosen, the program attempts to open the file requested by the user. If it is opened success- ▶

IF YOU MAKE A MISTAKE, YOU'LL HEAR...

THUNDER!™

FOR YOUR ATARI ST MACINTOSH AND AMIGA

"One of the most impressive programs we've seen... If you write, you need THUNDER!"
-ANALOG COMPUTING MAGAZINE

THE REAL TIME SPELLING CHECKER THAT WORKS WITH:

- Word Processors
- Personal Productivity
- Management Programs
- Telecommunication Programs
- Educational Programs
- Data Base Programs
- Finance Programs
- and many other programs!

ONLY
$39.95* For The ST
$49.95* For The Mac and Amiga
NOT COPY PROTECTED

THUNDER! is so much more than just a spelling checker! **THUNDER!** is also a document analyzer and a quick typist abbreviation expander!

BATTERIES INCLUDED™

BATTERIES INCLUDED, an ITM company, 30 Mural Street, Richmond Hill, Ontario, Canada, L4B 1B5, (416) 881-9941, Customer Information (416) 881-9816. If you can't find this product at your local retailer, you may order it direct from us at the full suggested list price plus $5.00 for postage and handling. For product orders please call 1-800-387-5707 (U.S. only). With all Batteries Included products you can always have the latest version of your program by returning the original disk and $10.00. Write to us for our full color catalog of products for the APPLE, APPLE MACINTOSH, ATARI, ATARI ST, COMMODORE, COMMODORE AMIGA AND IBM SYSTEMS. (C) 1986 BATTERIES INCLUDED. APPLE, APPLE MACINTOSH, ATARI, ATARI ST, COMMODORE, COMMODORE AMIGA, AND IBM are registered trademarks respectively of APPLE COMPUTERS INC., ATARI CORPORATION, COMMODORE BUSINESS MACHINES INC., AND INTERNATIONAL BUSINESS MACHINES INC.

* ALL PRICES SHOWN ARE IN U.S. DOLLARS. RETAILERS MAY SELL FOR LESS.

CIRCLE 006 ON READERS SERVICE CARD

fully, the program attempts a first-stage IFF read using the **iff_rd1()** function. The program passes the file's handle (**fhand**) to the function along with a 10-entry MFDB array (**workmfdb**) which the function will set up according to the block's size.

The **iff_rd1()** function returns a value of zero if the first-stage IFF read was successful. If the value is negative, an error has occurred. The error numbers and meanings are found in the **#defines** of the IFFRTNS.C file.

After **iff_rd1()** executes successfully, **Syspal**[] and **Gempal**[][] will be filled with the color values found in the IFF file. The MFDB array you specified in the **iff_rd1()** function call will be filled with the data for the bit-block *except* for the first two array entries. These entries, you should recall, contain the address of the memory form block, which the IFF Reader routines cannot determine. This is up to you, and we'll see how to do it in a moment.

The routine makes reading and writing IFF files a quick and painless task for the ST programmer.

The first thing IFFREAD does is check to see if either the width or height of the image is too large for the ST screen. If so, the MFDB values for these settings are adjusted to the maximum size of the ST screen. Excess size conditions can occur if you try to read a large monochrome image into a low-resolution screen (since monochrome images are 640X400 pixels and low-resolution images are 320X200 pixels). This operation will truncate the block to a usable size.

Next, we set the number of bitplanes in the form to the same number as the resolution that is in use. If the form has excess bitplanes, they are ignored. In the demonstration program, this is accomplished by:

```
workmfdb[6]=scrnmfdb[6];
```

If the user requested that the block be loaded into the screen (**formtype==1**), the address of the memory form is set to that of the current screen, which is in the **scrnmfdb**[] form entries [0] and [1]. To read the block into the screen, we must also make sure the block's width parameter (**workmfdb**[4]) matches that of the screen.

If the block is to be read into a memory form separate from the screen (**formtype==2**), we have to do a little more work. Specifically, we need to ask the operating system to allocate enough RAM for the bit-block's form. Fortunately, we can allocate just the right amount of RAM required for the bit-block—which can be as few as two bytes or as many as 32000 bytes. This prevents wasted memory.

To determine the amount of RAM needed to hold a bit-block, we do the calculation:

```
blocksize=MFDB[3]*MFDB[4]*MFDB[6]*2;
```

This value is then used as input to the GEMDOS **Malloc()** (memory allocation) call, which sets aside the RAM we need and returns the address of the allocated RAM. If the allocation is successful, we plug the RAM's address into locations [0] and [1] of the **workmfdb**[] array. The rest of the MFDB is left as is.

Now that the MFDB is set up with the address of the memory form and the number of bit planes we intend to use, we call **iff_rd2()**. This function actually reads the bit-block into the area we defined via **workmfdb**[]. If the read is successful, a zero is returned, otherwise, a negative value is returned (see the error code **#defines** in the IFFRTNS.C file).

After the phase 2 IFF file read has executed, the file we opened should be closed, since we're done with it.

If it is necessary to set the colors to those of the block, you can do it in two ways, as shown in the example program. The **Setpallete()** call is the easiest, but you can also use the **vs_color()** function to do the job. The result is the same in either case—it's a matter of personal preference. In the IFFREAD demo, we go ahead and set the colors in both ways so that you can see the block as it was stored.

If the block was read into the screen RAM, no further action is necessary, since the block is in view already.

If the block was read into a memory form other than the screen (**formtyp==2**), we'll have to copy it to the screen to view it. To do this, we set up the **blit**[] array with the coordinates of the block in its source form in **blit**[0] through **blit**[3]. We also set up the coordinates of where we want the block to be copied in the destination form (the screen) in **blit**[4] through **blit**[7]. For proper results, make sure the source block size and the destination block sizes are the same. IFFREAD copies the block to the screen so that it is centered horizontally and vertically. The actual instruction to ▶

TIME BANDIT

by Bill Dunlevy & Harry Lafnear

Listed as the #1 arcade game in CSS's top ten!

Blast the evil creatures and escape with the Treasure. Visit 16 different lands, each with 16 levels. Find the Artifacts and conquer the Gates of Time! Over 3,000 screens in all!

Includes 3 built-in text-adventures, and even has a Save Game feature! The conquest of Time and Space awaits you. .

(Color monitor or television required) $39.95

Cards $39.95

8 Ball $39.95

MAJOR MOTION

by Philip MacKenzie & Jeffrey Sorensen

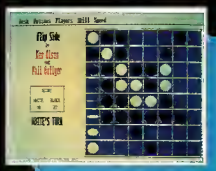
Hunt enemy spies in this thrilling driving game. Cars try to smash you off the road, and the enemy helicopter is never far behind. .

Race down highways, curves, ice, and even a waterway with islands and deadly reefs!

Force enemy cars off the road, or use your machine-guns. Missiles, smoke screens and other special weapons help you to survive.

(Color Monitor or television required) $39.95

Animator $39.95



Flip Side $39.95

These and other fine programs for your Atari ST. Ask for our latest catalog!

Dealer inquires welcome • Visa and Mastercard accepted • Add $3.00 shipping and handling to each order.

MichTron

576 S. Telegraph, Pontiac, MI 48053
Orders and Information (313) 334-5700

CIRCLE 064 ON READERS SERVICE CARD

do the bitblt operation is **vro_cpyfm()**, described earlier. It uses the **workmfdb**[] array for the source form, **scrnmfdb**[] for the destination form and the **blit**[] array for the coordinates to copy. Try changing the coordinate values to see how the bitblt operation works.

After you have viewed the loaded bit-block image, press any key to return to the Desktop.

WRITING BLOCK FILES

The IFFWRITE.C file on the START disk is the C source file for the IFFWRITE.PRG example program. With it, you can load any DEGAS-format picture and save portions of the picture to disk in a standard IFF-format block file. These files will be compatible with DEGAS Elite's block-handling capability and any other program that supports the IFF-format block files.

This program is quite straightforward. It is similar to IFFREAD in many places, so we'll cover mainly the important differences.

The program begins like IFFREAD by opening a virtual workstation, setting up the screen's MFDB, and so on. It does, however, perform an extra setup function. In order to prepare to write one of our block files, we must have a section of memory reserved to hold up to a screenful of data—or 32000 bytes. This is done with the **Malloc()** function. We ask the system for 32000 bytes of RAM, and it returns the starting address of the RAM. If this fails, the program informs the user with a dialog. Otherwise, we set the pointer **workarea** to the address of the RAM buffer.

Like IFFREAD, IFFWRITE displays the file-selector dialog. This time, however, it wants a DEGAS-format picture file as input, and automatically searches for a file with a .PI1, .PI2, or .PI3 extender, depending on the current system resolution. After the user enters the filename, the picture file is read into the current screen memory for the user to see. The 16-entry INT array **picpal**[] contains the color information for the picture.

After the picture appears, the user points the mouse to the upper-left corner of the area to be saved, presses and holds the button, dragging it to the lower-right corner of the block. A "rubber box" appears to define the rectangle that will be saved.

When the button is released, the program calculates the size of the bit-block and uses **vro_cpyfm()** to copy the block to the 32000-byte work area that was allocated at the start of the program. Before the bitblt operation can be executed, however, the program must set up the MFDB for the rectangle. This MFDB will be used to tell the IFF write routine where the block is and what its format is.

Now the program displays a second file selector dialog, this time expecting a block file name. When entering the filename, be sure to add the .BL1/.BL2/.BL3 extender so that you can identify the resolution of the block file later.

After the block filename is entered, the program creates the file and instructs the IFF write routine, **iff_wrt()**, to write the file. This is a simple one-stage process. The programmer furnishes the handle of the file that is opened as output, along with the MFDB for the block, the color palette that is to be used for the block, and the block's resolution (0, 1, or 2). The IFF write routine does all the rest and returns a 0 to indicate success or a -1 to indicate failure. After the write call returns, the program closes the file and the write process is complete! What could be easier?

An interesting side-benefit of this program is that if you "grab" the entire screen with the rubber box and write it, the resulting block file is directly usable by all Amiga paint programs that support IFF, making the program a handy DEGAS-to-IFF converter! Aren't standards wonderful?

FINAL NOTES

Remember: though these routines are flexible and will allow you to use any bit-block image regardless of resolution, loading an image with more bit planes than are used in your particular graphics mode may make the image unusable. If in doubt, just force the user to use .BL1, .BL2 or .BL3 files, depending upon the resolution. This way, the images will always be usable.

Using the IFF file-read routines shown in this example, you can load many different bit-block images into separate areas of computer memory for use by your own programs. Whether you are an ST developer or an involved programming hobbyist, these routines should prove valuable for using bit-block files in your own programs. ■

REFERENCE:

EA IFF 85 Standard for Interchange Format Files from Jerry Morrison, Electronic Arts. Available through Commodore-Amiga, 1200 Wilson Drive, West Chester, PA 19380.

GRAFCON ST, Universal Graphics Converter by Patrick Bass, Antic Magazine, July 1986, pp. 67-72.

GEM VDI Manual, Digital Research, Inc., pp. 6-1 through 6-8.

Atari ST GEM Programmer's Reference by Norbert Szczepanowski and Bernd Gunther, Abacus Software, pp. 157-161.

COMPUTER CREATIONS

YOUR ATARI 520ST SOFTWARE SUPPORT CENTER

ACTIVISION
- Hindshadow.....33
- Borrowed Time.....33
- Hacker.....29
- Music Studio.....39
- Paintworks.....47

ANTIC
- C.O.L.R. Object Editor...23
- Maps and Legends.....27
- Macro Assembler.....60
- Lattice C.....114
- Disk Doctor.....23
- A-Calc.....45
- Meta Pascal.....75
- CAD-3D.....38
- A-Ram.....15
- A-Disk.....27
- GST C Compiler.....60
- GST-ASM.....45
- Expert Opinion.....60
- Flash.....30
- Star Struck.....19
- Red Alert.....19

ATARI
- DB Man.....75
- DB Master.....34
- CP/M Emulator.....34
- Home Planetarium.....34

BATTERIES INCLUDED
- Degas.....26
- Timelink.....33
- Thunder.....26
- I/S Talk.....52
- Homepak.....33

HIPPOPOTAMUS
- Hippoword.....59
- Hippoconcept.....59
- Hippoemplate.....33
- Hippo Disk Utilities...33
- Hippo Backgammon.....27
- Hippo Spell.....27
- Hippo Randisk.....23
- Hippo Computer Almanac..23
- Hippo Jokes & Quotes....23
- Hippo Art I.....27
- Hippo Pixel.....27
- Hippo Fonts I.....27
- Hippo Laser.....47
- Hippo Eprom Burner....119
- Hippo Sound Digitizer..119
- Hippo Vision B & W.....119

HYCHTRON
- Kissed.....26
- DOS Shell.....26
- OPT.....33
- The Animator.....26
- Personal Money Mgr.....33
- Utilities.....39
- M-Disk.....26
- Mudpies.....26
- Softspool.....26
- Flip Side.....26
- Calendar.....19
- Hi-Term.....33
- Gold Runner.....26
- Time Bandits.....26
- BBS.....33
- Cornerman.....33
- Mighty Mail.....33
- Lands of Havoc.....13
- Cards.....26
- Business Tools.....33
- Logo.....33
- Major Motion.....26
- Mission House.....26

ARTWORX
- Strip Poker.....25
- Hole in One Golf.....20
- Mailmat.....12
- Pegoammon.....12
- Bridge.....19

EPYX
- Winter Games.....26
- Rogue.....26
- Temple of Apshai.....26
- World Games.....26
- Championship Wrestling.....26

HABA
- Business Letters.....29
- Wills.....29
- Hippo "C" Compiler.....49
- Habawriter.....39
- Phonebook.....39

INFOCOM
- Ballyhoo.....26
- Cutthroats.....26
- Deadline.....33
- Enchanter.....26
- Witchhiker's Guide.....26
- Infidel.....29
- Mind Forever Voyaging....29
- Planetfall.....26
- Seastalker.....26
- Sorcerer.....29
- Spellbreaker.....33
- Starcross.....33
- Suspect.....29
- Suspended.....33
- Wishbringer.....26
- Witness.....26
- Zork I.....26
- Zork II.....29
- Zork III.....29

PENGUIN/POLARWARE
- Crimson Crown.....26
- Go-Topos.....26
- Sword of Kadash.....26
- The Coveted Mirror.....26
- Transylvania.....26

SIERRA
- Kings quest II.....33
- Ultima II.....39
- Winnie the Pooh.....19
- Black Cauldron.....26
- K/Q Hint Book.....6
- B/C Hint Book.....6
- Donald Duck.....19
- Cash Disbursements.....67

TELARIUM
- Amazon.....34
- Nine Princes in Amber.....34
- Fahrenheit 451.....34
- Dragonworld.....34

TIMEWORKS
- Datamanager.....59
- Swiftcalc.....59
- Wordwriter.....59
- S/P Financial Planner.....59

UHISON WORLD
- Printmaster.....26
- Art Gallery I.....19
- Art Gallery II.....19

X-LENT
- Typesetter.....25
- Rubber Stamp.....25
- Music Box.....32
- Mecafont.....25

OSS
- Personal Pascal.....49
- Personal Prolog.....49

OTHER ST HITS
- Action Pak.....32
- Alternate Reality.....19
- Bratcacus.....33
- Easy Draw.....89
- Final Word.....Call
- Financtat Cookbook.....36
- Flight Simulator II.....Call
- Fontwriter.....27
- Forbidden Quest.....27
- Gateway.....34
- Hex.....27
- Homework Helper.....34
- Infinity Machine.....23
- Jet.....Call
- LDW-2000 Clock.....33
- Leaderboard.....27
- Macro Desk.....27
- Macro Manager.....49
- Mean 18.....27
- Mercenary.....19
- Mindwheel.....29
- Modula-2/ST.....59
- H Vision.....27
- PC Intercom.....84
- Phantasie.....27
- Silent Service.....26
- Soundwave SW-1.....33
- Spiderman.....14
- ST-Key.....14
- ST-Talk.....29
- Sundog.....27
- The Pawn.....30
- Treasure Island.....27
- Typing Tutor.....23
- UCSD Pascal.....59
- Ultima III.....39
- Universe II.....47
- VIP.....89
- Zoomracks.....54

PRINTERS

STAR MICRONICS
- LV-1210.....189
- HX-10.....259

PANASONIC
- KX-P 1080.....199
- KX-P 1091.....239
- KX-P 3131.....259
- KX-P 1092.....319
- KX-P 1592.....429

MODEMS
- Supra 300 ST.....59
- Supra 1200 ST.....149

ST ACCESSORIES

- Disk Cleaning Kit.....14
- Dust Covers.....7
- Monitor Stand.....12
- House Pad.....8
- Shanner Planner.....29
- The Easel.....14
- Disc Directory.....20
- Pocket Pak.....10
- The Library.....34
- Speed Pad.....18
- Disk File/30 Micro.....10

ATARI 520 ST HARDWARE: CALL

Package #1

Atari 520 ST Computer & SF 354 Disk Drive

Package #2

Atari 520 Computer, SF 354 Disk Drive and SC 124 Monochrome Monitor

Package #3

Atari 520 ST Computer, SF 314 Disk Drive and SC 124 Color Monitor

Package #4

Atari 520 ST Computer, SF 354 Disk Drive and SC 1224 Color Monitor

Package #5

Atari 520 ST Computer, SF 314 Disk Drive and SC 1224 Color Monitor

CALL FOR PACKAGE PRICES

SF 354 SS/DD Disk Drive.....Call
SF 314 DS/DD Disk Drive.....$209
SM 124 Monochrome Monitor....Call
SC 1224 Color Monitor.....$329
SHD 204 20 MG Hard Disk .....Call

3½"	Sony	Sony
Bulk	SS/DD	DS/DD
10-29	**1.69 ea.**	**2.49 ea.**
30+	**1.59 ea.**	**2.29 ea.**

3½"	Sony	Sony
Box (5)	SS/DD	DS/DD
2-6	**11 Bx.**	**16 Bx.**
7+	**10 Bx.**	**15 Bx.**

Atari ST GEM Reference
Atari ST Internals
Atari ST Machine Language
Atari ST Tips & Tricks
Atari ST from Basic to C

ST BOOKS

$ 16 ea.

Atari ST Basic Training
Atari ST Graphic & Sound
Atari ST Logo
Atari ST Peeks & Pokes
Atari ST for Beginners

To order call TOLL FREE
1-800-824-7506
ORDER LINE ONLY

No extra charge for MasterCard or Visa

COMPUTER CREATIONS, Inc.

P.O. BOX 493 - DAYTON, OHIO 45459

For information, order inquires, or for Ohio orders (513) 435-6868

VISA® MasterCard

Order lines Open 9:00 a.m. to 8:00 p.m. Mon.-Fri.; 10 a.m. to 4:00 p.m. Sat (Eastern Standard Time) Minimum $15 per order. C.O.D. (add $3.00). Please specify computer system. Call toll free number to verify prices and availability. Prices and availability are subject to change without notice. We ship C.O.D. to Continental U.S. addresses only! Please include 4% shipping on all Hardware orders (min. $4.00). Software and accessories add $3.00 shipping and handling in continental U.S. Actual freight will be charged outside U.S. to include Canada, Alaska, Hawaii, Puerto Rico and APO. Ohio residents add 6½% sales tax. Canadian orders add 5% shipping. (min. $5.00). All other foreign orders, please add 15% shipping. (min. $10). For immediate delivery send cashier's check, money order or direct bank transfers. Personal and company checks allow 3 weeks to clear. School purchase orders welcome. Due to our low prices, all sales are final. NO CREDITS. All defective returns must have a return authorization number. Please call (513) 435-6868 to obtain an RA# or your return will not be accepted for replacement or repair. FOR YOUR PROTECTION WE CHECK FOR CREDIT CARD FRAUD. We do not bill until we ship.

CIRCLE 012 ON READER SERVICE CARD

DISK INSTRUCTIONS

Welcome.

All programs and program listings in this issue are on your START disk provided in a special envelope bound into the magazine. (If you purchased the $4 version of START, without disk, you can still purchase the disk by sending us the bound-in card or mailing $10.95 plus $2.00 for postage and handling, to START DISK, 524 Second Street, San Francisco, CA 94107.)

Use a pair of scissors to open your disk envelope along the outside vertical edge. Place your START disk into drive A and double-click on the disk icon to see its contents. Please refer to your ST owners manual if you are uncertain about proper Desktop procedures.

Your START disk contains nine folders. Each folder corresponds to a particular START article. To open a folder and reveal its contents, double-click on the folder's icon. Before running a program, we recommend that you read its related article and transfer its files to another disk. Also, it's a good idea to back up your START disk before use.

COMPRESSED FILES

Some of the files on the START disk are in a compressed format. You can identify them by a "Q" as the second letter of the filename extender (e.g., XLISP.TQP). These files are unusable unless first decompressed with the Un-Squeeg program included on your START disk. We chose the Squeeg program from Quack Computer Company as the most efficient compressor of ST files. Only the Un-Squeeg program (the decompressor) has been included on your disk. Please note that this program is owned and copyrighted by the Quack Computer Company and is not to be considered part of the public domain. The latest Squeeg/Un-Squeeg package, which includes the compression program, is available for $24.95 from:

Quack Computer Company
257 Robinson Avenue
Bronx, New York 10465
(516) 689-8738

DECOMPRESSING

To decompress a file, first transfer both the compressed file and UNSQUEEG.PRG to another disk. When run, the Un-Squeeg program will create a second, decompressed file which may be more than twice the size of the compressed one. Un-Squeeg will *not* alert you if you run out of disk space, so make sure you have plenty of room on your destination disk.

After transfering the files, double-click on UNSQUEEG.PRG and a file selector box will pop up requesting the file to decompress. Un-Squeeg automatically searches for filenames with "Q" as the second letter of the extender. Select a file to decompress and click in the OK box. Un-Squeeg responds by asking for a destination path. At this point, it is easiest just to select the default (drive A) by clicking in the OK box. The file will be decompressed and written to disk with the original, unsqueezed filename (e.g., XLISP.TQP will be decompressed and written to disk as XLISP.TTP).

As an example, we'll step you through the decompression of the XLISP program:

1. Drag UNSQUEEG.PRG to a blank, formatted disk in drive A, then open the XADVISOR.STQ folder by double-clicking on it, and drag XLISP.TQP to drive A. (If you plan to go on and use the programs from the Chabris AI article, this might be a good time to drag all the files within the XADVISOR.STQ folder to drive A. There will be plenty of room.)

2. Double-click on UNSQUEEG.PRG from drive A to run the Un-Squeeg program.

3. The Un-Squeeg program will begin with a file-selector box requesting a file to decompress. Click and highlight the file XLISP.TQP, then click in the OK box.

4. Un-Squeeg will respond by asking for a filename to write the decompressed file to. The filename will default to XLISP.TTP on drive A. For this example, select the default by clicking in the OK box.

5. The file XLISP.TQP will be decompressed and written out as XLISP.TTP. When the Un-Squeeg program is done, it will query you for another file to decompress. Click in the CANCEL box to exit to the Desktop. The decompressed XLISP file, XLISP.TTP, will now be on drive A and ready to run.

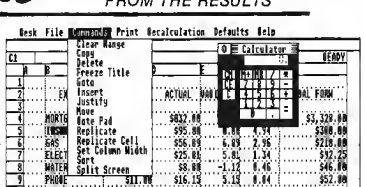
ADDITIONAL INFORMATION

Detailed information on batch files, program compilation, and program assembly is contained in the README.TXT file on your START disk. All Programs were compiled and tested on a 520 ST with TOS in ROM, using the Atari Developer's Kit ▶

EZ Calc™

PUT EZ CALC™ TO WORK FOR YOU AND BENEFIT FROM THE RESULTS

EZ CALC™ is a fully implemented **GEM™** based spreadsheet for home and business use. This is by far the most powerful spreadsheet available for the price. Better yet, all commands are mouse controlled for speed and ease of use. EZ CALC™ also uses **less memory** than other spreadsheets for the ST, leaving more room for your data and formulas. **If you've never used a spreadsheet before, you'll be amazed how easy** EZ CALC™ **is to learn and use. The experienced user will love the speed of a mouse controlled spreadsheet.**



ONLY $69.95

FEATURES

- 300 columns by 999 rows
- Extensive use of **GEM™** windows
- All commands are under mouse control
- Built in 10 keypad calculator
- On-line help windows (No commands to memorize)
- Built in sort routine
- Developed exclusively for the Atari ST
- 10 macros controlled by the function keys
- Split-screen capabilities
- Note Pad

GEM is a Trademark of DIGITAL RESEARCH, INC. **EZ CALC** is a Trademark of ROYAL SOFTWARE.

MOUSE CONTROL

Extensive use of the **GEM™** windows make **EZ CALC™** a fast, extremely easy-to-use spreadsheet. With over 50 commands available from the mouse, the ease of use is unsurpassed. Imagine being able to move or copy an entire column of figures with a simple mouse control.

CALCULATOR

EZ CALC™ includes an easy to use 10 key calculator that can be pulled down at anytime and operated either by mouse or keyboard. With the point of a mouse, the results of the calculation can then be transferred to the cell of your choice.

NOTE PAD

EZ CALC™ lets you attach a personal note of up to 4 lines to any cell. The cell is then highlighted to remind you there is a note attached. For example, you could attach a note to the insurance cell of your personal finance spreadsheet reminding you that the cell applied only to car and home insurance. The note pad can be pulled down at any time.

★ **FULL FEATURED DEMO OF EZCALC AVAILABLE FOR $5. REFUNDABLE WITH PURCHASE.** ★

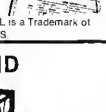
Help Calc™ Only 24.95

For The Atari ST™

- 11 preprogrammed templates for use with **EZ CALC™** or **VIP Professional™**
- **load-and-go** and these templates will take the work out of tedious spreadsheet setup.

Templates include:

- Check Register
- Depreciation schedules
- Investment Portfolio Analysis
- Name & Address directory
- Home Inventory
- Loan Amortization Schedules
- Personal Finance Statement
- and more VIP PROFESSIONAL is a Trademark of VIP TECHNOLOGIES



ST STAND

ONLY **$39.95**

+ $5.00 Min. Shipping & Handling

Custom made just for the ST, beautifully finished stand to hold your ST monitor, 2 disk drives, a modem, disk files, ETC...

IF YOU DON'T HAVE OUR CATALOG . . . . . YOU'RE MISSING OUT ! ! !

Over 1500 items for your Atari

Become a preferred customer and save $ $ $ $

ORDER YOURS TODAY!

- SPECIALS
- CLOSE-OUTS
- BARGAINS
- REVIEWS
- WHAT'S NEW
- PLUS MUCH MORE!

SEND $2.00 to get one year subscription to our catalog

DELUXE DUST COVERS

Deluxe Leather Grain

PROTECT YOUR INVESTMENT

Custom fitted, attractive leather brown color:

7.95 • **KEY PAD**—ATARI CX85 • **PRINTER**—ATARI 1020 • **RECORDER**—ATARI 1010

8.95 • **COMPUTERS**—400/800, 600/800/1200XL 65/130XE • **DISK DRIVES**—ASTRA 1620, ATARI 1050 ST DRIVES ST HARD DRIVE, INDUS GT PERCOM 88S1/88SPD RANA 1000 TRAK • **PRINTERS**—ATARI 1025/1027 AXIOM SLP/GLP OKIMATE 10 • **RECORDER**—ATARI 410 • **MODEM**—ATARI 1030

9.95 • **COMPUTER**—ATARI 520/1040ST • **PRINTERS**—ATARI 825 AXIOM 550 CITOH/PROWRITER 8510 CPA 80/EP-150 EPSON MX/RX/LX/FX80 W/ & W/O TRACTOR FEED GEMINI/STAR SG 10/X PANASONIC 1091/1092 RITEMAN II/PLUS

12.95 • **MONITORS**—B/W/RGB MONITOR TEKNIKA MJ-10 • **PRINTERS**—EPSON FX 100/185

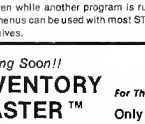
HELPMATE™ Only $29.95

For The Atari ST™

HELPMATE ST™ INCLUDES:

- 10 Key calculator
- Appointment calendar with alarm functions
- Telephone/name index

All in one program



The best part is that HelpMate stays "hidden" in memory until needed, and then can be called up for use, even while another program is running. The pull down menus can be used with most ST programs or by themselves.

Coming Soon!!

INVENTORY MASTER™

For The Atari ST™

Only $99.95

INVENTORY MASTER™ is a powerful, Inventory control and Report generation program. It will do more than just keep track business inventory, such as: detailed report generation, fast and easy data retrieval, versatile data entry, takes the work out of decision making, plus much more.

COMPUTER PALACE

OPEN M-F 9-6 Sat 10-4 (Pacific Time)

2160 W 11th Avenue Eugene, Oregon 97402

USE YOUR CREDIT CARD & CALL

Toll Free 1-800-452-8013

★ **ORDERS ONLY, PLEASE** ★

There's never a penalty for using your credit card!

For Information, Call (503) 683-5361

Prices subject to change without notice.

CIRCLE D13 ON READER SERVICE CARD

SHIPPING INFO. Minimum $2.90 Ground $4.75 Air. Actual Cost depends on weight. Call (503) 683-5361 for information. **WARRANTY INFO.** Everything that we sell is warranted by the manufacturer. If any item purchased from us fails to perform properly when you receive it, call us at (503) 683-5361 so that we can assist you. No returned merchandise accepted without authorization. Defective software will be replaced with another copy of the same program, otherwise no software is returnable.

• 2 Day Air Shipping **AVAILABLE** •

DISK INSTRUCTIONS

THE DISK

UNSQUEEG.PRG
Double-click on this to decompress a file.
README.TXT
Text file of information not included on the Disk Instructions page.

Parsers, Raams, Objects and Tralls, page 54

ADVENTURE.STQ

ADVENTUR.TOS
Double-click on this to run Adventure.
ADVENTUR.CQ
Compressed Adventure C source code.
ADVENTUR.BAT
Adventure batch file.
EASYC.H
Easy C defines.
GAME.H
Adventure C equates.
PDEFINE.H
Miscellaneous C defines.

Which C far Me?, page 62

BENCHMRK.STQ

BENCHMRK.C
Alcyon, Megamax, and Lattice source code.
BENCHMRK.GST
GST C source code.
BENCHMRK.BAT
Dhampstone batch file (for Alcyon C).

FujiBaink! page 109

FUJIBOIN.STQ
(See sidebar, pg. 113).
FUJIBOIN.PRG
Actual FujiBoink! runtime program. Be sure you have created all the data before running this program
FUJIBOIN.CQ
Compressed FujiBoink! C source code.
FUJIDRAW.PRG
Draws the 32 images of the fuji, creates FUJIDRAW.D8A; run in low-res.
FUJIDRAW.CQ
Compressed FujiDraw C source code.
FUJISHAD.PRG
Calculates the reflective properties of the fuji; creates FUJISHAD.D8A
FUJISHAD.CQ
Compressed FujiShade C source code.
GETTITLE.PRG
Loads a neochrome picture, cuts out the title screen, and creates TITLE.D8A.
GETTITLE.CQ
Compressed GetTitle C source code.
PEND.PRG
Combines the three data files into FUJIBOIN.D8A.
PEND.CQ
Compressed Append C source code.
FUJISTUF.SQ
Compressed interrupt routine assembler source code.
FUJI.BAT
FujiBoink! batch file.
TITLE.NEO
Neochrome picture.

Swapping Art with Other Computers, page 39

IFF.STQ

IFFREAD.PRG
Reads in and displays an IFF data block (.BL_); (see pg. 44).
IFFREAD.CQ
Compressed IFF Read C source code.
IFFREAD.BAT
IFF Read batch file.
IFFWRITE.PRG
Reads in a Degas picture file and allows you to create an IFF block. (see pg. 48).
IFFWRITE.CQ
Compressed IFF Write C source code
IFFWRITE.BAT
IFF Write batch file.
IFFRTNS.O
IFF Routines object code to link in with your own applications
IFFRTNS.CQ
Compressed IFF Routines C source code.
IFFRTNS.BAT
IFF Routines batch file
BEE.BL1
Sample low-resolution IFF block.
CHART.BL2
Sample medium-resolution IFF block.
JET.BL3
Sample high-resolution IFF block.

The Amazing MouseTrap page 19

MOUSE.STQ

MOUSTRAP.PRG
MouseTrap program; copy into an AUTO folder on your boot disk. If no AUTO folder exists, create one with the New Folder option from the Desktop menu
MOUSETRAP.SQ
Compressed MouseTrap assembler source code
MOUSTRAP.BAT
MouseTrap batch file.
MOUSEGAG.PRG
Mouse practical joke; copy into an unsuspecting victim's AUTO folder.
MOUSEGAG.SQ
Compressed Mouse Gag assembler source code.

Reversi Desk Accessary, page 12

REVERSI.STQ

REVERSI.ACC
Reversi Desk Accessory; copy into the root directory of your boot disk and turn on the computer with the disk in the main drive.
REVERSI.CQ
Compressed Reversi C source code.
REVERSI.BAT
Reversi batch file.

Structured IO, page 77

STRUCTIO.STQ

STRUCTIO.TOS
Double-click on this to see the Structured I/O Demo. Warning: creates a file EMPLOYEE.DAT and expects a non write-protected disk in drive A (see pg. 78).
STRUCTIO.CQ
Compressed Structured I/O C source code.
STRUCTIO.BAT
Structured I/O batch file.
STRUCTIO.INP
Link68 input line for batch file.

Probing the FDC, page 96

TRAKREAD.STQ

TRAKREADTOS
Track Reader program (see pg. 107).
TRAKREAD.SQ
Compressed Track Reader assembler source code.
TRAKREAD.BAT
Track Reader batch file.

The AI Apprentice with XLISP, page 22

XADVISOR.STQ
(See sidebar, pg. 26).

XLISPTQP
Compressed XLISP program. In order to use any of the LISP programs, this file must be decompressed using the Un-Squeeg program.
CHESS.RUL
Xadvisor knowledge base for chess tactics.
HIRING.RUL
Xadvisor knowledge base for faculty hiring.
INIT.LSP
XLISP initialization file.
PPRINT.LSP
Pretty Print LISP utility.
XADVISOR.LSP
Xadvisor LISP program. ■

Regent

REGENT BASE
A FULL FUNCTION RELATIONAL DATABASE
Regent Base's procedural language make it a natural for handling any of your small business needs. Modules are available for Invoicing, Accounts Receivable, Checkbook Balancing, General Ledger, etc.
Regent Base is a relational database written specifically for the Atari ST. Don't settle for simple clones of IBM products. Regent Base is easy to use and state-of-the-art!

REGENT SOFTWARE
7131 Owensmouth, Suite 45A
Canoga Park, CA 91303
(818) 883-0951

ATARI ST

CIRCLE 100 ON READER SERVICE CARD

NEW

from the creators of MacFORTH

Multi-Forth
for the Atari ST

A Professional Development Language for the 520 and 1040ST

- **Turnkey Compiler - No Royalties**
- **GEM and TOS Support**
- **Multi-tasking**
- **Floating Point**
- **CompuServe Support - GO FORTH**
- **Built-in Assembler**
- **Use Standard Text Files**
- **400 Page User's Manual**
- **Compatible with other CSI Forths**

Shipping Now - $149 Intro Price

1-800-FORTH-OK

Creative Solutions, Inc.

4701 Randolph Rd., Suite 12
Rockville, MD 20852
in MD 301-984-0262

MacFORTH & Multi-Forth are trademarks of Creative Solutions
Atari ST is a trademark of Atari Corporation

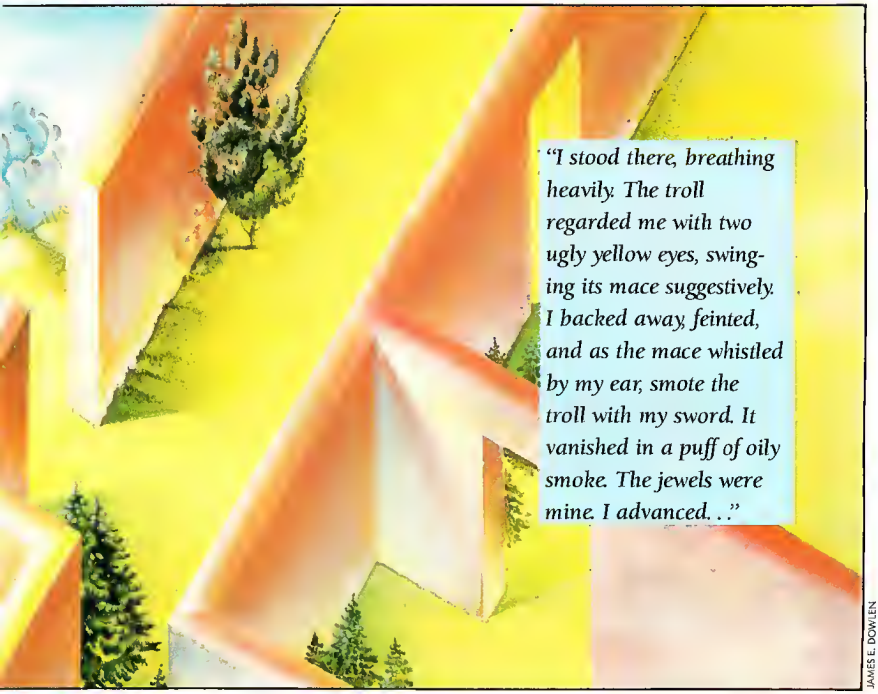
CIRCLE 014 ON READER SERVICE CARD

PARSERS, ROOMS, OBJECTS AND TROLLS

THE ADVENTURER'S TALE

by Alex Leavins

The outhor of the 8-bit Atori text odventure, Wombots, reveols how to create text adventures in C. Includes on onolysis of mopping, porsers ond object monipulotion—plus o "mini," five-room text odventure which you will discover within the ADVENTUR.STQ folder on your START disk.



JAMES E. DOWLEN

Most readers will recognize this as part of a text adventure, an interactive book, in which the computer acts as guide, referee, scorekeeper and adversary in an alternate–and often quite complex–world, a world that exists only inside the computer itself. The rules are not always clear, and combat (if there is combat) can prove fatal. But don't worry–there's always the Reset button.

A typical adventure-game interplay between computer and player might read like this:

You're standing in a small room; a sputtering torch is on the wall.
−−> TAKE TORCH
OK, you now have a torch.
−−>EXAMINE TORCH
It's just your basic torch.
−−>NORTH
You're standing in a long corridor, with exits to the south and west.
−−>EAST
You can't go that way.
−−>WEST
This is the end of the corridor. You can go east from here. There are some jewels here.
−−>TAKE JEWELS
Taken.
−−>EXAMINE JEWELS
They're beautiful gemstones, agates and diamonds.

The −−> is a prompt telling the player to enter a command. The uppercase words represent the player's input, and the computer's responses are shown in lower case. In our example, the computer prints a location description, "This is the end of the corridor. . .," followed by the −−> and waits. The player then types TAKE JEWELS. The computer tries to recognize the input and take the appropriate action, in this case, taking the jewels. ▶

OK, we've got a basic understanding of what an adventure game is. Now, how do they work?

BUILDING A UNIVERSE CAN BE TOUGH

We're going to create a simple text adventure in C, which you will find on your START disk within the folder ADVENTUR.STQ. Refer to the Disk Index page for further information. We have no bells and whistles in this program, no GEM, no graphics – just the essentials: five rooms, some objects, and a two-word parser. Once you understand this structure, you can expand upon it, and perhaps create your own Infocom-style game.

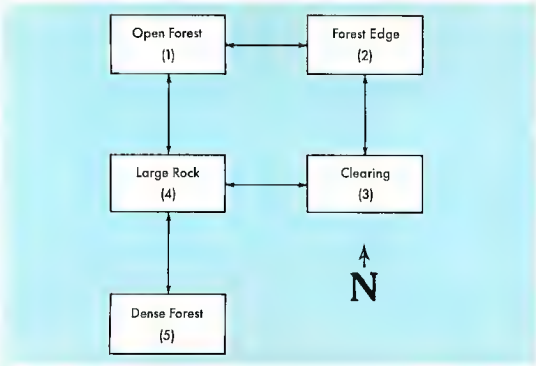


FIGURE 1

Let's ignore for a minute just how our program will interact with the player. This is not a trivial matter, but before we can address it we must first determine what the game itself is going to do. Every adventure game includes a number of discrete locations, known as *rooms*. Room is a generic term, referring to any distinct location in the game. It needn't be inside a building, indeed it needn't even be enclosed. If it's a unique, specific location that a player can travel to, then it's a room. All of the rooms in our game, taken all together, make up the game's *universe*. Here's the universe for the simple adventure game that we'll be building (see *Figure 1*).

Here we have five rooms, each of which leads to at least one other room. (Rooms aren't very useful if you can't get to them.) Going east from the Open Forest room (room #1) will take us to the Forest Edge room (room #2). Similarly, going south from the Large Rock room (room #4) will take us to the Dense Forest room (room #5). We can summarize all of this as in *Figure 2*.

So, north from room #3 leads to room #2. South leads nowhere, as does east, and finally, west will take us to room #4. The rows give us the direction we wish to move in, the column headings give us the room we're currently in. If we're in room #4 and want to move north, we look in the NORTH row, under "room #4," and discover the number 1, which is the Open Forest room. Ah, but what if we don't find a room number, but a zero? Easy. Zero means for that room, that direction leads nowhere. By the way, you'll note that going north from room #1 leads to – room #1! This often happens in adventure games; rooms will double back on themselves, lead to rooms way on the other side of the universe, or other, nastier surprises.

If you'll look at the code of ADVENTUR.C on your START disk, just before **main()**, you'll notice that I've translated the above table to a series of integer arrays, containing room values. We have a **north**, **south**, **east**, and **west** array, corresponding to the four directions. Movement for our program is just as simple as the table; **north[1]** is the room we'll reach if we walk north from room #1.

This is implemented in the routine **move_us()**. This routine takes a direction (n, s, e, w) and a current position (the room the player is in), and determines what room (if any) the player ends up in. It does this by a simple **switch** on the desired direction, which then routes the current room into the appropriate directional array. The capitalized versions of **NORTH**, **SOUTH**, **EAST** and **WEST** are defines contained in the file GAME.H, which represent 1, 2, 3, and 4, respectively. After the **switch**, the value of **i** is either the room that the player is now in, or zero. If it is zero, we know the player can't go in that direction, and a simple test followed by a **printf** statement lets the player know this. If the move is valid, we show the player the new room and any objects that might be there.

Ah, objects. Haven't mentioned those yet, have I?

THE CARE AND FEEDING OF OBJECTS

The universe of rooms is one building block of adventure games. Objects are another. Briefly, an object is any item that can be seen, taken, used, or interacted with in any other way. Objects in sophisticated adventure games can have many properties. For example, not all

objects seen in a room may be "takeable," while some takeable objects may not be revealed in the original room description. Complex, eh? Don't worry, in our adventure game, an object is easily spotted by the phrase:

"There is an XXXXX here."

where XXXXX is the object in question. In our game there will be four objects: a book, an axe, a coin and a stick. Each of these will be in a different room, and can be acted upon in several simple ways. We will be able to *take* an object, *drop* it, *examine* it, *read* it, even *throw* it. Each of these commands is implemented by a separate routine. Obviously, **take_object()**, takes objects. Each of the other routines is similarly named, except **exam_object**, which is shortened to prevent a seven-character label-name conflict.

The basic procedure for each of these routines is the same. First, determine the command requested by using the **switch(cmd_index)** statement in **main()**. Next, find out if the word following the command is a valid one, with **split_word()**, which separates the succeeding word from the input string and puts it into a test array, and **scan_objects()**, which tests the word in the test array against the objects that the game recognizes. If the object is one that the game knows, the program attempts to process it, using the appropriate **xxxx_object** routine. If the object is *not* recognized, then the game prints a short message and waits for the next input.

So far, I've avoided the issue of *how* the computer recognizes words. This is the third major aspect of adventure games, and is known as the *parser*.

PARSERS, OR WHAT'S THAT WORD?

We now have some understanding of the low-level routines that process game procedures, such as the taking of objects. But how do we get from the player typing "TAKE BOOK" to actually *taking* that book?

Obviously, a computer doesn't "know" words, it can only process numbers. We need a way of translating the user's input into simple numerical values—or tokens—that the game can use to determine what happens. This translation process is performed by the game's parser, which takes words and sentences, breaks them down, analyzes them, and tokenizes them by assigning a unique number to each command. The array **commands[]** and the array **objects[]** (just before **main()**) contain all the words that our game will recognize:

COMMAND	TOKEN	OBJECT	TOKEN
NORTH	1	BOOK	1
SOUTH	2	AXE	2
EAST	3	COIN	3
WEST	4	STICK	4
TAKE	5		
DROP	6		
THROW	7		
READ	8		
EXAMINE	9		
INVENTORY	10		
LOOK	11		
QUIT	12		

Notice that these are the actual words that the game recognizes; they are literal strings. The computer recognizes these words by their token numbers, which are actually the positions of the words within the **command[]** or **object[]** arrays. Notice also that the file GAME.H contains each of these words in a define statement:

```
#define NORTH 1
#define SOUTH 2
#define etc.
```

Here the words are actually labels that are replaced with the corresponding numerical values at compile time. So there are two versions of the same word, both related to the same number, but seen differently by the program. Why do we do this? There's a very good reason—it's just a little confusing at first.

Let's go back to our parser. We want it to accept a word or two of input, such as NORTH, then translate that into a numerical value, such as 1, so that the program can do something like:

```
if(command_value == 1)
{
   . . . /* move the player north */
}
```

From:	room #1	room #2	room #3	room #4	room #5
North	1	0	2	1	4
South	4	3	0	5	0
East	2	0	0	3	0
West	0	1	4	0	0

FIGURE 2

Take a look at that if statement. Without the added comment, the code isn't particularly obvious, is it? What does the 1 mean, anyway? But now suppose we do this:

```
#define NORTH 1
.
.
.
if(command_value == NORTH)
{
   . . . /* move player north */
}
```

This makes things much clearer to someone reading the code, and makes the code a whole lot easier to write. We can now reference, in the code, the actual command tokens that were created by the parsing language. We can build the entire adventure game, and use the command and object names that the game itself recognizes, without having to reference the token values of a command or object.

OK, enough of tokens, objects, and rooms. Let's get to the heart of the matter: the parser.

PARSING AGAIN, OR ENGLISH MADE SIMPLE

The player has just typed "TAKE BOOK" and is eagerly awaiting a response. OK, what's the first word? What do you mean, what's the first word? It's obvious. TAKE is the first word. Ah, but it's not obvious to the computer, which sees language as just a string of bits. But it's easy to define a "word" for the computer. A word, in this case, is any text string, followed by a blank space. So, in the following sentence:

TAKE THE GHSTS AND THE BOOK

TAKE, THE, GHSTS, AND, THE, and BOOK are all words to the computer—even though GHSTS is no word that I know. With our simple definition of "word," scanning the input string and separating it into discrete words becomes simple. It is implemented in the routine **split_word( )**, by scanning the input array, **parser**[], and looking for either a blank space, or the end of the array (since the last word in a sentence won't have a space after it). Once the blank space is found, everything is copied from the start of the array up to that space, into the array **test_word**. Now, shift the remainder of the array *after* the space, to the start of the array, so that the next time we examine the array the next word will be at the beginning.

We now have the first word of the player's command, TAKE, in the array **test_word**[]. We next pass it to **scan_commands( )**, which will test it against the list of known words. You'll note that the array **commands**[] contains all our command words, with each one separated by an asterisk. The asterisk is used as a delimiter so that **scan_commands( )**can tell where one word ends and the next one begins. We use the pound sign (#) to tell the routine that there are no more words.

The **scan_commands( )** routine itself is very simple. Read a command word into the test array **showword**[], then do a string comparison between it and the word in array **test_word**[]. If the result is zero, the strings are identical and the routine returns the index of the word, which is the position the word occupies in the array **commands**[]. If the result is not zero, the routine tests to see if we've exhausted our word list. If not, then the word index is incremented, and the next word is read into the array **showword**[]. If we have exhausted the list of words in **command**[], we exit the routine with a returned value of zero, which tells the main routine that we didn't recognize the word.

PUTTING IT ALL TOGETHER

We now have all the pieces we need to put together an adventure game. Let's walk through the entire procedure for a single command, to get a feel for how it all works. We'll examine how the adventure game processes the command:

TAKE BOOK

Starting at the top of the **WHILE (1)** loop in **main( )**, we first execute **get_input( )**, which simply waits for the player to type a string into the array **parser**[]. Note that if the player simply presses [Return] without typing anything, the game displays a brief "Say what?" message, and recycles through the keyboard-input routine.

The player types TAKE BOOK, and **get_input( )** returns with the string TAKE BOOK in the array **parser**[]. Now, we call **split_word( )**, which breaks the first word, TAKE, out of the array **parser**[], and puts it into the array **test_word**[]. It then shifts the string in **parser**[] over, so that **parser**[] now contains the string BOOK. Now **test_word**[] contains the string TAKE. Next, we call **scan_command( )** which compares the words NORTH, SOUTH, EAST, WEST, and TAKE to the word in **test_word**[]. On the fifth pass through the loop it finds a match in the word TAKE, and returns the number 5, which is put into the variable **cmd_index**.

Now the program tests the value of **cmd_index**. Zero would mean the player had typed a command that the program didn't recognize, and the message "I don't understand that command," would be displayed before it returned to the top of the **WHILE (1)**

loop. However, in our case **cmd_index** equals 5, so the program passes it to **switch(cmd_index)**, where **case TAKE** will be executed. (Remember, we've defined the word TAKE to be 5, in GAME.H.)

The first thing the program does in the **case** statement is call **split_word()** again. Now the word BOOK is shifted into the array **test_word**[1], leaving the array **parser**[1] empty. Then the program calls **scan_objects()**, which does for objects what **scan_commands()** does for commands. It scans through the list of objects in the game to match them against the test word. In this case, **scan_objects()** tests the word BOOK, finds a match, and returns a value of 1, which is put into the variable **obj_index**. The program then tests the value of **obj_index** to see if it is a recognized object. If not, the program will print "I don't know what that is," and exit to the top of the **WHILE (1)** loop. In our case, it *did* recognize the word, and so passes its index to **take_object()**, which performs the actual taking of the object.

Notice that up until this point the only thing our parser has done is to break down the input of the player, and process it to find out what to do with it. No "game" actions have taken place. It's only now that the program can invoke any of the game mechanics themselves.

TAKING THE OBJECT

The program has passed to **take_object()** the index of the item the player wants to take. Note in GAME.H that we've defined:

```
#define BOOK 1
#define AXE 2
#define COIN 3
#define STICK 4
```

The first statement in **take_object()**is an **IF** statement, which says:

```
IF (inventory[obj] EQ ON) THEN
   printf("You've already got
   it!\n");
```

But what's inventory? Since the program needs some way to determine, at any time, what the player is carrying, we implement this in a simple fashion with the array **inventory**[]. The value of nth element of **inventory**[] tells the program if the player is carrying the nth object. If the value is zero, or OFF, the player isn't carrying the object, but if the value is 1, or ON, the player is. Thus, to see if the player is carrying the BOOK (which has an index value of 1), the program looks at **inventory**[1].

The **IF** test now becomes clear: If the player is already carrying an object, it can not be taken again. Assuming the player is not carrying the BOOK, the next statement,

ELSEIF (where[obj] NE position)

is executed. The array **where**[] works exactly the same way as the **inventory**[] array, except that the **where**[] array defines what room an object is currently in. So if **where**[3] equals four, the third object (in our game, the COIN) is currently in room #4 (in our game, the Large Rock room). If some value of **where**[] is zero, the related object is in the player's possesion. Note that this overlaps the inventory function. We could, if we wanted, use the **where**[] array to define both the placement of objects in the universe, and objects currently in the player's possesion. But it is much simpler, conceptually, to have two separate arrays.

The variable **position** defines the current position of the player. If the object the player is trying to take is not in the same room,the player can hardly get it. But if we assume the book is in the room, the program sets **where[obj]** to OFF, indicating that the object is no longer in a room, then sets **inventory[obj]** to ON, indicating that it's in the player's possession. The program then prints a message that the player has taken the object and exits the routine. Back in **main()**, we've reached the **last** statement in **case TAKE:** and we're done with our example. Ta-da!

ENHANCEMENTS

Adventure games are great fun to build, but can be frustrating because there's always still one extra feature that's just too good to leave out. Here are a few ideas for improving our little adventure game (aside from enlarging its universe). I'm sure you can come up with many others.

- Increased vocabulary— Obviously an adventure game is only as good as the words that it can handle. Some adventure games handle a few hundred; others, such as Infocom's, handle more than a thousand. A bigger vocabulary also creates a much better illusion of reality.
- Better semantic and syntactic handling of English— Our adventure game uses a simple verb/noun parser limited to either one- or two-word commands. Wouldn't it be nice to have it recognize such things as:

DRAGON, GIVE THE RARE BOOK TO THE ANGRY TROLL

All of this is possible, but would require at least a thorough understanding of the elements of English sentences. If you're serious about building a better parser, I recommend that you first acquire further information on the syntactic relationships of words (see *Reference*, below). As an example of the kinds of things you'll have to consider when designing your parser, here are two sen-▶

SEARCHING FOR ***ADVENTURE***

Looking for ST adventure? There's plenty out there. Because of the huge amount of available memory, the ST is a natural for labyrinthine adventure games. Most of the following products are adventures with graphics, although some (such as Infocom and the Synapse series) are strictly text

Any Infocom Game
(you can't go wrong)
Infocom Software
125 Cambridge Park Drive
Cambridge, MA 02140
(800) 262-6868
$39.95-$44.95
CIRCLE 220 ON READER SERVICE CARD

Essex
Brimstone
Mindwheel
Synapse/Broderbund
17 Paul Drive
San Rafael, CA 94903
(415) 479-1170
$44.95
CIRCLE 221 ON READER SERVICE CARD

Borrowed Time
Mindshadow
Activision
P.O. Box 7286
Mountain View, CA 94039
(415) 960-0410
$44.95
$49.95
CIRCLE 222 ON READER SERVICE CARD

Crimson Crown
Transylvania
Polarware/Penguin
830 4th Avenue
Geneva, IL 60134
(312) 232-1984
$19.95
CIRCLE 223 ON READER SERVICE CARD

The Black Cauldron
King's Quest II
Winnie the Pooh*
Donald's Playground*
Sierra On-Line
P.O. Box 485
Coarsegold, CA 93614
(209) 683-6858
$49.95
$39.95
$24.95 (* For younger adventurers)
CIRCLE 224 ON READER SERVICE CARD

The Pawn
Firebird Licensees
P.O. Box 49
Ramsey, NJ 07446
(201) 934-7373
$44.95
CIRCLE 225 ON READER SERVICE CARD

Gateway
Action Software
69 Clementina St.
San Francisco, CA 94105
(415) 974-6638
$39.95
CIRCLE 226 ON READER SERVICE CARD

tences that are semantically identical (they mean exactly the same thing) but syntactically quite different:

GIVE THE BOOK TO THE ALIEN
GIVE THE ALIEN THE BOOK

Both sentences have the same result: the alien is given the book. But the direct and indirect objects are exactly reversed, and the word TO is missing in the second sentence.

• More complex objects– Wouldn't it be nice to give each object a variety of characteristics, such as weight, size, visibility (objects might appear only after a player has done some other task first), "takeability" (which would define under what circumstances you could take an object), and many other things. Something like this could perhaps be implemented using a structure. For example:

```
struct object = {
                    int weight;
                    int size;
                    int take;
                    int visible;
                  };
```

might define the weight, size, takeability and visibility of an object. Then we could define objects simply, in terms of the generalized structure. Every object would have the same set of possible properties and we could build a more complex game without a great deal more work.

• Frames– A concept from the artificial intelligence field, a "frame" is a set of default conditions. Suppose we made the default condition for all objects as takeable. We would then only have to define–or flag–those objects in the game which were *not* takeable. By establishing a default condition covering the majority of situations, we eliminate a lot of code we would have had to apply to each individual object. Applied throughout the game, this concept can save a great deal of coding time. But keep in mind that, once programmed,

default behavior cannot be changed.

• More text– Wouldn't it be nice to describe every room with a whole screenful of rich, imaginative text? Instead of printing a simple one line string for each room, we could open up a text file corresponding to the room number and read in as much text as we wanted. We could apply this concept to everything, not just rooms, so that a great many actions would yield interesting, non-trivial results. Or, how about placing pictures instead of text in our files?

• Independent characters– How about characters that walk around the universe independent of you and help or hinder your progress, depending upon your interactions with them?

How about ., but it's pointless, I could go on for days. Suffice it to say the ultimate adventure game has not yet been written. Up to now, we have seen only the fledgling first steps of a whole new artform. With patience, persistence, and a little bit of luck, you might enlarge the five rooms in our sample program and help shape the future of adventure gaming. ■

REFERENCE:

- *Zork and the Future of Computurized Fantasy Simulations* by David P. Lebling, BYTE magazine, December 1980, vol. 5, number 12.
- *Syntactic Structures* by Noam Chomsky, Mouton, The Hague, Netherlands.
- *Foundations of Syntactic Theory* by Robert P. Stockwell, Prentice-Hall, New Jersey.

EZRAM™ 520

512K Memory Upgrade for the Atari520ST®

Featuring the **EZ**Temp™ Soldering Guide

Upgrade Your 520ST to a Full Megabyte of RAM

- Increase spreadsheet and database capability
- Dramatically improve RAM disk capacity for enhanced I/O operations

Designed for Simple Installation

- Features the **EZ**Temp™ solder template. All the soldering occurs on the template not at the RAM chips. Eliminates chip stacking.
- Clear, easy to follow, illustrated installation instructions.

Free Software **Mada in the U.S.A**

- Memory check diagnostic software and additional accessory programs included.

S.L.: $199.00 **6 Month Warranty**

terrific PERIPHERALS SM

See your Dealer or call us at (617) 232-2317 Brookline, MA 02146

EZRAM and **EZTemp** are Registered Trademarks of Apex Resources, Inc. Atari & Atari 520ST are registered trademarks of Atari Corp

CIRCLE 121 ON READER SERVICE CARD

I.B. Drive provides the link

between IBM PC/XT/AT and Atari ST computers. Just insert a 5¼" diskette in either computer as drive B:. Read, write and format in either computer

- *Plug-in compatible with Atari ST*
- *Built-in power*
- *40 track 360K capacity*
- *80 track 720K capacity (optional)*
- *Complete, ready to run*

$269.95

I.B. Computers
1519 S.W. Marlow Ave.
Portland, OR 97225 U.S.A.
503/297-8425

We carry the complete line of Atari hardware and software

For further details see your local Atari dealer or call I.B. Computers.
Dealer inquiries invited

IBM and PC/XT/AT is a registered trademark of International Business Machines Corp.
Atari and ST is a registered trademark of Atari Computer Corp.

CIRCLE 040 ON READER SERVICE CARD

WHICH C FOR ME?

START'S FIRST C COMPARISON

by Arick Anders and Michael Bendio

START compares the four available ST C languages: Alcyon, Lattice, Megamax, and GST—including revealing benchmarks plus a critical analysis of each compiler's methods, documentation, editors, and extras.

Comparisons are odious—and often misleading. This is especially true of programming languages. No matter what yardsticks are used, certain elements of the test are bound to be unfair to various implementations. But language comparisons provide necessary and valuable guidelines to both the serious developer and the dedicated hobbyist.

One way to compare C's is to first determine the features of the "perfect" compiler, then find a way of evaluating these features objectively. Easier said than done. A number of tests, called "benchmarks," have been developed to quantify these features. The best known—the Sieve of Eratosthenes—measures a compiler's ability to manipulate several variables rapidly. But benchmarks often do not accurately measure what they appear to measure. For example, the following code was once used to measure a compiler's ability to manipulate pointers:

```
strlen(s) /* returns the length of s */
char *s;
{
        char *p;
        for( p = s; *s != '\0'; s++) ;
        return(s - p);
}
```

This really tests the efficiency of the **for** loop. The amount of time spent doing pointer manipulations is trivial compared to the time spent executing the loop.

Distinguishing between a compiler that generates good code and one that uses an efficient library is important. If you rarely use library calls, a poor compiler with a fast library is not the one for you.

DHAMPSTONE

Of all the generic benchmarks published over the years, the best we have seen is Dhampstone, by Jack Purdum. Dhampstone, first published in the February, 1986 issue of Computer Languages, is actually several benchmarks in one. It avoids library calls and does a good job of testing specific features of a compiler. And all of Dhampstone's subroutines return a value which helps ▸

KYLE HOUGH

the programmer verify if the compiler generates correct code. We added timing routines for the GST C test and modified the program to run on the Atari ST. (Look in the BENCHMRK.STQ folder on your START disk for the Dhampstone source code.) In all cases, each compiler returned the correct values from the subroutines, so no indication of the returned "magic" numbers is included in our charts.

The peculiar name for this benchmark is a tongue-in-cheek reference to two other benchmarks. The Whetstone benchmark is a well-known measure of floating-point performance from the mainframe world, and the Dhrystone benchmark, written by C. Weicker, and published in CACM (Communications of the ACM magazine), is based on a study of keyword frequency in programs written in several different languages. The Dhrystone benchmark attempts to show the distribution frequency of these keywords. Purdum, however, made a similar study of keyword distribution directed specifically at C in some 15,000 lines of C code from the Digital Equipment Corporation User's Group in Marlboro, Massachusetts. He found significant differences in C keyword distribution from Weicker's study. The Dhampstone benchmark reflects Purdum's findings for C's keyword distribution. It includes tests of string handling, arithmetic involving integers, unsigneds, longs, doubles, and disk I/O. The unsigned test also checks a compiler's ability to handle recursion by finding Fibonacci numbers (a series in which each number is the sum of the previous two, i.e., 1, 1, 2, 3, 5, 8, 13, etc.). While these are simple trials, they do a good job of testing a compiler in its critical tasks.

DOODLE

To test GEM compatability, we used DOODLE, the paint program included with the Developer's Toolkit. This is a good example of a typical GEM application in terms of size and usage. It also revealed some interesting violations of programming standards. For example, Alcyon C will allow a pointer to be assigned to a nonzero constant or to a variable. Kernighan & Ritchie specifically forbid this. With the pickier compilers, such as Megamax, it causes portability problems. It also uses long integers as pointers.

The greatest portability concern we encountered resulted from the lack of a standard size for an integer variable. Lattice and GST use a 32-bit integer; Alcyon and Megamax use a 16-bit integer. Both the new ANSI standard (see below) and Kernighan & Ritchie define an integer size as the "most convenient" size for a particular machine. Historically, integers have been the same size as pointers, but there is no real reason they should be, especially since making integers the same size as pointers tends to promote programming abuse. Since the 68000 is a 16-bit processor and 32-bits requires two memory fetches, we have to agree with Alcyon and Megamax in choosing 16-bits for the default size.

Among these products, there are astounding differences in the compile and link times.

The quickest way to categorize the available ST compilers is to decide if you need a simple compiler or a full GEM developer's kit. Alcyon and Megamax offer GEM developer's kits which include resource construction sets and full GEM documentation. Lattice and GST are compiler packages. They can be used to produce GEM applications, but with more difficulty.

Among these products, there are astounding differences in the compile and link times, and the size of the executable programs. One reason GST's size on the Dhampstone is so much larger is that it always includes code for windowing. Megamax's times are so much quicker because it is a single-pass compiler and doesn't have to wait on disk I/O for intermediate files.

In evaluating each compiler, we have tried to identify the best features of each and express its basic "feel." We have left most of the performance evaluation to the benchmarks (see our accompanying charts). The quantitative differences between the compilers are quite distinct.

ANSI

Recently, the American National Standards Institute (ANSI) established the X3J11 committee to define a standard for the C language. In addition to clarifying many ambiguities in C as defined in "The C Programming Language" by Kernighan and Richie, the committee has added several new features to the language which are beginning to show up in new releases of C compilers. These include the "void" data type for functions which don't return significant values, enumerated data types, structure assignment, the use of "unsigned" as a modifier for other data types, and perhaps most importantly, function prototyping.

A frequent source of errors can be traced to adding or deleting arguments to functions and failing to make corresponding changes to all of the calls to those functions. In the past, C compilers have blithely compiled such errors without complaint. On the first call with mismatched parameters though,

the program suddenly begins executing parameters or passing code—with disastrous results. Function prototypes allow the programmer to declare the number and type of a function's parameters, and the type of its return value. With this information the compiler can detect and warn of mismatched parameters. Lattice C implements all these ANSI extensions. Its function prototyping allows it to detect the wrong number of parameters, and it will complain if you pass a pointer to an **int** instead of an integer. However, it remains silent if you pass a **char** to an **int**; it would be nice if the type-checking here were stricter.

FLUCTUATIONS

As dynamic as the ST is, it is hardly surprising that, within the time this review was written, one C compiler had been withdrawn from the market and one was being introduced. Our review will cover the four compilers that are *currently* being marketed: Lattice, Alcyon, Megamax, and GST. Many people are familiar with Hippo-C from Haba Systems. We talked with Pat Merrifield from Haba and he informed us that Hippo-C is no longer being marketed by Haba Systems. Hippopotamus Software originally developed Hippo C for in-house use. Haba Systems purchased the rights from Hippopotamus to finish and market it. Haba subsequently released it, but after receiving a flood of error reports, withdrew it from the market.

On a more positive note, Mark Williams, of Mark Williams Company, manufacturers of Let's Write, indicated that he is developing a C compiler for the ST. Unfortunately it was not ready in June 1986, when this article was written. Mark Williams is well known in the world of minicomputers. He also has a compiler on the IBM PC which is one of the few available on a microcomputer with a source-level debugger. START will examine this compiler package when it becomes available.

LATTICE C

Anyone who has worked with C in the MS-DOS environment will recognize and welcome Metacomco's introduction of Lattice C to the Atari ST marketplace. Although Microsoft C is currently challenging Lattice C for market leadership on the IBM PC, it is probably safe to say that more applications have been written with Lattice C than any other MS-DOS C compiler. Thus, the Atari ST version of Lattice C will particularly interest anyone considering a port of an MS-DOS application to the ST.

Lattice's recent 3.0 release for MS-DOS was a major upgrade from the 2.x versions in two areas. It added the new ANSI C language extensions, and included an improved, more UNIX-compatible library. We were glad to see that the ST version included the ANSI extensions. This indicates to us that the compiler portion of the ST product is Lattice's newest and best. The library, however, is definitely the equivalent of the older 2.x MS-DOS library rather than the newer 3.0 release. This really amounts to saying that the library is merely very good instead of being excellent. The function index for this implementation has more than 130 entries covering memory allocation, I/O, utilities, string handling, and math functions.

DOCUMENTATION

Documentation for Lattice C consists of a paperback manual of about 250 pages. It includes a two-page table of contents, five appendixes, and an eight-page index. Two of the appendixes thoroughly document error messages and even suggest possible sources for the errors. The remaining appendixes include example programs, a function index, and documentation of the GEM VDI/AES interface. The manual itself contains sections on the editor, the linker, the language definition, and machine dependencies in the code generation. The bulk of the manual—a full half of it—is devoted to UNIX-style documentation of the library. Each library function has its own page with sections explaining its purpose, giving its synopsis and description, specifying its return value(s), and noting any relevant cautions.

A major weakness in the manual is its documentation of the VDI/AES interface, which is relegated to a 16-page appendix. Each entry includes a routine name plus a short phrase describing its action. Arguments for the routines are listed, but not the argument types or legal values. This appendix seems intended only as a guide to the heavily commented assembly-language source code for the VDI/AES calls, provided on the disk. Although the source provides much more information about arguments and how to use the routines, the information is still woefully inadequate by itself.

GEMDOS, BIOS, and XBIOS calls allow access to machine-specific parts of the ST operating system. These vital in-▸

terface routines are documented by Lattice with only a single page each. But it's hard to fault Metacomco for this poor documentation when the blame should really be placed at Atari's door. VDI, AES, and the GEMDOS, BIOS, and

We were glad to see that the ST version of Lattice included the ANSI extensions.

XBIOS should be thoroughly documented in a series of manuals available from Atari at a reasonable price. To make this information available only to developers, and only with $300 worth of bundled software is unprofessional and counterproductive. IBM has set a standard for thorough and freely available documentation that Atari would do well to follow.

USE

Creating an executable program with Lattice C is a two-step process. First, you invoke the compiler-driver, **LC**, which opens a dialog box prompting you for a single file name and any command line specifiers you wish to use. **LC**, in turn, executes the two phases of the compiler: **LC1**, which parses the source and outputs a quad file, and **LC2** which generates the code. The second step is linking, using the GST linker that Metacomco supplies. Again, you use a dialog box to pass a command line to the linker. The linker is identical to the one included in the GST C package.

Several compile-time options can be specified when running **LC**. You can redirect error and warning messages to a specified file by using **>filename**. The **-dsymbol** or **-dsymbol=value** options define the identifier "symbol." This can be used to trigger conditional compilation directives in the code (**#ifdef**, **#ifndef**, etc.). Although the default effective length for identifiers is eight characters, you can override this with the **-n** option which causes the compiler to use up to 31 characters to differentiate identifiers. Stack overflow bugs can be extremely difficult to diagnose. The **-p** option causes the code generator to insert a special instruction known as a "stack probe" at the entry of each function. If the stack overflows, this code detects it and aborts the program with a "stack overflow" message. After you're sure your code is clean, you can recompile without the stack probe option. This can save you hours of debugging. The **-i** option is useful for specifying in which directory the compiler should search for **#include** files.

EDITOR

The editor provided with Lattice C is serviceable. It's not mouse-oriented, nor does it support windows, but it does allow the basic text manipulation necessary for program editing in a straightforward way. The Control key handles the more frequently used commands. Other, extended commands, are typed on a command line at the bottom of the screen after pressing the Esc key. The default size of the text buffer is large enough to edit a file of about 60K. A command-line option is available for larger files.

EXTRAS

Unlike the GST and Alcyon compilers, Lattice C includes no assembler. Unlike the Megamax compiler, Lattice C includes no disassembler. It makes no provision for inspecting code generated by the compiler, nor for optimizing it by hand. The calling protocol for assembly routines is clearly documented, as is the protocol for calling a C function from an assembly language module.

Up to eight variables may be declared register variables. The compiler reserves four address registers for pointer variables and four data registers for simple variables. A peculiarity of this compiler is that integers are 32 bits long—the same size as longs. This creates relatively larger and slower code.

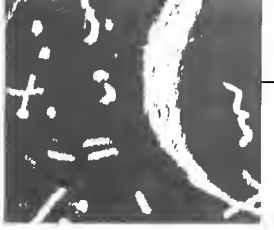
The compiler, linker, editor, and other necessary files take up about 293K of disk space. If you're using one single-sided drive, this doesn't leave much room for source. At least one double-sided drive or two single-sided drives are necessary to use this compiler effectively.

SUPPORT

Metacomco, a British company, supplies an address in Scotts Valley, California for customer support, as well as their home address in Bristol, England. We at first suspected the California address would be essentially a marketing center, but were surprised to find real technical assistance at the other end of the phone line.

CONCLUSION

Lattice C represents an excellent value for the price. The mature compiler and its compatibility with the foremost MS-DOS compiler makes it an attractive package. The documentation is clear and nicely presented. Compared to Alcyon C, its compile and execution times are very respectable.



ALCYON C

We bought our Atari STs with the intention of developing and selling software. At the time, the only software tools available were bundled in the Developer's Toolkit, available from Atari for $300. Although there are now alternate choices, the bona fide software developer who intends to bring a product to market will find the developer's package indispensible. The tools aren't the best, and the documentation leaves a lot to be desired, but they suffice to get the job done. Plus, Atari wants you to succeed in your product development; quite often they are able to assist you with your marketing efforts.

DOCUMENTATION

The developer's package includes a compiler and assembler from Alcyon; a linker, debugger, and various utilities from Digital Research, Inc. (DRI); and an editor, telecommunications package (Kermit), and other utilities from Atari. The documentation consists of a six-inch-plus stack of over 2,000 double-sided loose sheets. The principal parts of the documentation are a several-hundred-page GEM Programmer's Guide, a "Hitchhiker's Guide to the BIOS," the "Long-Awaited Line 'A' Document," a GEMDOS manual, and CP/M68K documentation for the compiler, linker, debugger and utilities. The CP/M68K documentation is indexed and includes extensive tables of contents, but it also contains much that pertains only to the CP/M68K environment and must be disregarded. In short, the information is (probably) all there, but there's a lot of chaff with the wheat, and you'll find yourself doing a lot of sifting.

The package includes sample programs plus two excellent examples that help answer many GEM-related questions. ACCSKELC and APPSKELC are, respectively, skeletons for desk accessories and for GEM applications. Add your code and you're on your way. The hard part of hooking up with GEM is done for you.

It's been our experience that working with new equipment means using Xeroxed, poorly-organized documentation, tools that don't work well together, and having to figure out a lot of things on your own. That certainly is the case with the developer's package. It was perhaps justifiable a year and a half ago but, unfortunately, it is still true today. It is time we start to see typeset, indexed, organized documentation, and tools with more polish than is currently being offered by Alcyon, DRI and Atari.

USE

The Alcyon compiler consists of four pieces: cp68, the preprocessor; c068, the parser; c168, the code generator; and as68, the assembler. A batch-file utility invokes these components.

Alcyon supports many of the same compile-time options that the Lattice compiler does. **−dsymbol**, and **−idir** work the same way. **−e** specifies the use of IEEE floating point format. This supports single and double-precision floating point. **−f** specifies Motorola's "Fast Floating Point" format which supports only single precision.

The Developer's Toolkit provides an alternate environment for program development. If your tastes run more to traditional shells, you may prefer COMMAND.TOS to GEM. Invoke it, and you will see a bare **[a]** prompt. The commands are not documented, but if you are familiar with UNIX and MS-DOS, they are what you would expect. We use COMMAND.TOS frequently with the Developer's Toolkit. Constantly growing and shrinking windows can become obnoxious after a while, and dialog-box command lines seem to be a clumsy way to start up a compiler.

Two additional tools are unique to this compiler and are invaluable. **nm68** is a utility similar to the UNIX **nm**. It takes a ".O" file as its input and prints its symbol table. This is handy for resolving undefined function errors during link. Another unique facility is SID, which stands for Symbolic Interactive Debugger. It is an assembly-level debugger that supports basic operations such as single-stepping, disassembling, examining and changing registers or memory, and setting breakpoints. It doesn't have a mini-assembler built in, and it's not symbolic, but it can be very useful at times.

The Alcyon compiler supports two of the five ANSI language extensions previously mentioned. It allows structure assignments and the use of "unsigned" as a type modifier. It parses function prototypes, but makes no use of them for type-checking.

Alcyon C allows a maximum of eight register variables. Three registers are available for pointer variables and five for other register variables.

EDITOR

The editor supplied with the Alcyon package is called Micro EMACS. If you've used the full EMACS on a large machine you know that it is an endless source of delight. It's fully customizable, extensible, and comes with an enormous amount of on-line help and documentation. Of this, Micro EMACS retains a minimal, non-customizable, non-extensible subset. It's not a bad editor; its major fault is the lack of a search and replace function. A minor fault is the awkward Esc-V key sequence used to page through the text. On the positive side, Micro EMACS can ▸

edit text in several (non-GEM) windows, but it doesn't support the use of the mouse. An informal survey found that most of those who spend the majority of their time editing vs. word processing, preferred to use the Micro EMACS editor.

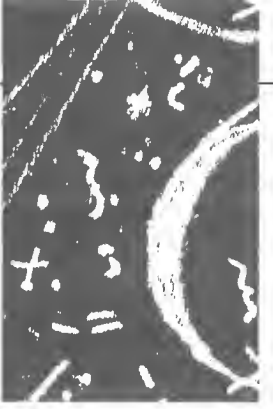
The minimal working set of compiler, linker, batch utility, editor, and necessary libraries for Alcyon C comes to 455K of disk space. By splitting compiler and linker files and juggling disks, it is *possible* to work on small programs using Alcyon with one single-sided drive, but not really practical. Indeed, the lengthy compile and link times for this package make any floppy-based system almost unbearable. The only real alternatives are a large RAMdisk, or a hard disk.

SUPPORT

To really take advantage of Atari's support with this package, you will need a modem and a CompuServe account. Atari maintains a CompuServe SIG dedicated to registered developers. Here you'll find frequent bulletins with the most-asked questions and their answers, announcements, copious source code to study, additional tools and utilities, and direct access to Atari's technical and marketing support.

CONCLUSION

There's a lot to love and a lot to hate about this package. The support is unparalleled. The documentation has more information than is available from any other single source. And there are more tools and utilities that come with this compiler than with any other. On the other hand, the documentation is poorly organized and contains significant amounts of irrelevant and erroneous material. Also, the compiler and linker are by far the slowest of the four reviewed. The principle advantage to Alcyon C and the Developer's Toolkit is its support.



MEGAMAX C

Megamax is hoping to lure software developers away from DRI and the Alcyon package. With this intent, they are offering most of the features that the Alcyon development package has, plus some nice extras. The heart of the Megamax package is a one-pass compiler. If you are as tired of working in a batch mode with your C compiler as we were, then you will really enjoy working with a one-pass compiler. Compiles that took ten or fifteen minutes now compile in one or two. Megamax claims that it compiles ten times as fast as the Alcyon compiler. With smaller programs, we have noticed about a five to one ratio. When compiling DOODLE, the difference was incredible!

DOCUMENTATION

After wading through 4,000 pages of developer's documentation, or looking up brief, inadequate GEMDOS function descriptions, the Megamax documentation is a joy to behold. It is clearly written, and typeset by laser printer. Each function call is on a page by itself and the user is almost never sent thumbing through to the manual to find explanations. Generally, where the information could reasonably be reproduced, it was inserted where it was needed—even if this meant copying sections of the manual verbatim. They do have a small problem with the VDI section: All the function calls in the index are off by one page. But they are there! They have even reprinted all of the "H" files for easy reference.

An important consideration for any professional ST developer is the quality of the GEM hooks. We could find no GEM hooks, BIOS or XBIOS calls that Megamax did not support. Just as important, they have thoroughly and clearly documented them. The package includes over 300 pages of documentation with only one function per page. The table of contents is accurate and the index is a sight for sore eyes. Without question, the documentation is the best available to an ST software developer.

USE

The Megamax shell places you in an environment very similar in look and feel to the GEM Desktop. Several drop-down menus give you the ability to compile, link, execute, change libraries etc. Thus, without leaving the shell, you can take a program from source code to execution.

In conjuction with the shell, Megamax has created their own dialog boxes which offer some nice features unavailable with standard GEM boxes. Megamax's boxes provide considerably more information and you can scroll through an entire directory by holding down the mouse button on the scroll arrows. To avoid cluttering your screen, the dialog box only displays the applicable files. For example, while in the compiler, only the "C" files are displayed. Unfortunately, while the shell remembers what disk drive you are using, it doesn't remember what directory you are in. This means that anytime you need to compile or edit, you have to walk the program down from the root directory. The path length that you are able to type in is somewhat limited as well. A

hard disk makes these limitations particularly annoying. But, overall, the shell is quite convenient to work with.

A feature of the Megamax compiler some developers may feel is a disadvantage is that it does not and cannot directly produce assembly source code. This speeds up the overall compile process tremendously. But, if you like to develop in C, then manually optimize the intermediate assembly source code, it can be quite involved. A disassembler is included, so compiling your code, then disassembling it is possible. But this whole process is annoying.

Megamax produces "pure," relocatable code, which means it separates the data and program code. Because they are separate, if you exceed the bounds in an array reference, you destroy data, but you don't destroy the program itself. However, this introduces the major drawback of the Megamax compiler: no single execution program segment can be over 32 kilobytes in size. This limits any given function to less than 32 kilobytes and requires that the programmer add compiler directives in the source code, telling it to break up the code into appropriate-sized blocks. The linker then resolves addressing problems between overlays by means of jump tables. Megamax is considering using this as a memory-management and overlay scheme at some future date. However, at this time, the entire program loads into memory. Except for adding the compiler directives to the source code, this whole process is completely transparent to the developer and the user.

While this is a holdover from the compiler Megamax developed on the Macintosh, it has some advantages. The first is that, within any given segment, the execution times are much better. It also means that if Atari comes out with a multi-tasking operating system, memory management is already built into the code. Since Atari is a UNIX licensee, this is something to keep in mind. The code is generated almost exclusively with PC relative addressing which reduces code size and increases code speed.

One of the nicest features of this compiler is the error log. If you have a program which takes forever to compile, you can get up and stretch your legs. Should an error occur, the compiler will save it to a file. When you get back, just check for the presence of an error file. We do have a quibble with this feature: If you have an error file left from a previous compile, an error-free compile will leave it intact. It would be less confusing if a clean compile erased the old file.

The linker is an intelligent linker; it only loads the external symbols and functions it needs, thus reducing code size. Because the 68000 has special addressing commands that take advantage of nearby addresses, the smaller code size can mean faster execution speed. When the linker scans for either a function or a symbol, it stops at the first incidence of the definition. Since the system library is loaded last, the user can redefine parts of a library and still access other parts of the same library.

Megamax can be used on one single-sided disk drive.

The Megamax linker is the slowest program of the package. To use the linker from the shell, you single-click the files that you want to link, then add them to the link-list window. Once you have added all the files you want linked, click the OK button and Megamax loads and links the program. The system library is automatically searched and is not even listed when the program asks what you wish to link. Therefore, unless you create additional libraries with the "librarian," you only need to explicitly link one file.

The librarian referred to above is an interesting additional feature. One of its principle functions is to facilitate modular compilation. Once you have a module debugged, you compile it and put it into a library module. Then you just link the library module with the development module at link time. Since the librarian resolves all local references, this not only reduces the amount of code requiring recompilation, but speeds up the linker. It also reduces the size of the library module.

Megamax can effectively be used on one single-sided disk drive: a nice, money-saving feature. The basic compiler, linker, librarian, system library, shell, some sample source code, etc., use less than one single-sided disk. Thus it takes up minimal storage on a hard disk, and the whole program will fit nicely onto a RAMdisk. Unhappily, portions of the Megamax files must remain on the top-level root directory, so you cannot place the whole system within a folder. This is inconvenient for hard disk owners. With a RAMdisk, the edit, compile, link and execute cycle is measured in seconds. Even with the standard single-sided disk drive and 512K of RAM in a minimal 520 ST system, the program performs elegantly.

Another particularly nice feature is that Megamax can handle, compile, and link a source file which is in a different ▸

disk drive than the compiler, linker and header files, without requiring any special contortions on the part of the programmer. Just single-click the disk drive you want to use and the shell lists all the applicable files on that disk. If you want to use them, activate them with a single-click, then click the OK button. If you are compiling, the compiler will produce a "O" file on the same disk as the source code. If you are linking, by default, the program will produce an A.PRG file on the same disk as the system. You can redirect it to a different disk or file name if you like. Unfortunately, it will not do multiple compiles or links. Nor will it compile and link in one command.

Megamax has abided by the Kernighan & Ritchie standards and added several extensions from the new ANSI standard. These include structure passing, structure assignment, and functions returning structures. The same member name can be included in more than one structure, and character constants can be both integers and longs. To allow programs larger than 32K, they also added the overlay mechanism mentioned previously.

EDITOR

The Megamax editor is a mouse-based editor that includes complementary keystrokes for the drop-down menus. Nonstandard use of the arrow keys was the worst problem that we encountered; the cursor keys move the window instead of the cursor. Only the mouse moves the cursor. Another major failing: It can't handle files greater then 32K. While this editor may be a little better than some of the keyboard-based editors available, it is quite poor and difficult to use. Along with the 32K compiler limitation, the editor is a weak point in the Megamax package. We do not feel that it is adequate for a professional environment and, in fact, do not use it. We have found the GST editor to work very well within the Megamax shell. For that matter, 1ST Word, which is free, will also work very well with the Megamax system.

EXTRAS

The Megamax package comes with several unusual additions, including: a code optimizer, a full resource construction set, a disassembler, a "make" utility, Megaroids (the arcade game bundled with the ST, written in Megamax C), and a set of example programs. Some of the example programs demonstrate writing directly to screen memory, how to format a floppy disk from within a program, use of in-line assembly, creating application skeletons and accessory skeletons, and how to create a dialog box. Additionally the source code for the support libraries is available for $50. This includes both disk and printed hard copy. A hard copy of Megaroids is available for $25.

The GST shell uses the GEM environment better than any of the other packages.

SUPPORT

We have called upon Megamax for technical assistance a number of times and invariably have found that the Megamax technicians not only knew their own compiler, but understood GEM! Having first written a C compiler for the Macintosh, they have some unique insights to the strengths and weaknesses of both systems and are a valuable asset to any programmer. Megamax has one full-time person dedicated solely to supporting their compilers. This may be contrasted with companies who rely on their compiler developers to also substitute as support. Megamax does not have toll-free phone number, but they do publish their phone number where it is plainly visible and usable. Since the first copy of the compiler that we received was a pre-release version, we called them a number of times. Without exception, they were very helpful.

Megamax has a reasonable upgrade policy. For $20.00 they will send you the latest version of the program and any necessary documentation.

CONCLUSION

At two hundred dollars, Megamax has priced their compiler beyond the casual programmer. But the serious developer will have a hard time finding a better C system. Megamax has concentrated on those areas where the Alcyon compiler is weakest, i.e., its interface, speed of execution and documentation. They have produced a very nice package. Admittedly, some will find the 32K program limit a major flaw in this package. But we have worked considerably with other C compilers on both minis and micros, and the only thing we miss with the Megamax compiler is a source-level debugger and a decent editor. It is a good, solid package that any programmer can use and enjoy.

GST

We had a good first impression of GST's C package. Its interface is smooth and the editor very similiar to 1ST Word, the word processor that comes bundled with the ST. Unfortunately, we soon felt very restricted and frustrated. This product is a very good prototype of a professional package. Unfortunately, at the time of this writing, it is too unfinished to be a serious contender for the professional developer.

Currently, it does not support some of the fundamental constructions of C. This is particularly limiting since many GEM calls require the address of a structure as a parameter. This doesn't mean that productive and professional products can't be produced with this package. In fact, according to their documentation, the compiler itself and all the other products from GST, including 1ST Word, were written either with this compiler or their macro assembler. However, in addition to problems with the language implementation, the GEM documentation accompanying GST C is not sufficient to produce a product ready for market. They refer you to the GEM manuals from the Developer's Toolkit. As previously mentioned, GST shouldn't be expected to reproduce the information on GEM, but compared to the other packages, the amount of GEM documentation is extremely limited.

DOCUMENTATION

The GST documentation leaves quite a bit to be desired. Their explanation of the various menu items in the shell is fairly thorough, but they fail to explain all of them clearly. It has no index, but the table of contents is quite good and it has a function summary showing the parameters that each function needs. Unlike most of the other systems, GST gives a brief explanation of each of the language constructions they have implemented to date. This serves to document the features of their implementation and it introduces the beginning programmer to the C language.

The accompanying system libraries are a reflection of the entire GST language implementation: an interesting combination of possibilities and missed opportunities. GST C has a nice standard C library and it is well supplemented. However, only a few UNIX-standard calls have been implemented. It is missing several low-level UNIX calls such as "open" and "close". In spite of not having these calls available, the file I/O is surprisingly quick. Since casts, structures, floating point and noninteger functions are not available with GST, there was no way to compile DOODLE, and we had to rewrite the benchmarks. The timer routines available with GST were only accurate and precise to within two seconds, which made several of the benchmark times meaningless.

Offsetting a rather strange standard library is a delightful supplemental GEM library that removes much of the tedium from the GEM interface by doing some of the housekeeping for you. As part of this approach, your program is automatically placed in a window. It is pretty startling the first time you write a "hello world" program and see it come up in a window with borders, scroll bars and the rest of the window niceties. GST's version of windowing calls are quite a bit easier to work with than the normal GEM calls. If portability is not a concern, this is a very nice feature. Depending on your application, you may not need or want the low-level routines that were not implemented. This library could justify using GST if your primary concern is developing a quick prototype and you do not want to spend a lot of time fighting with GEM.

USE

One of the reasons GST's limitations are so frustrating is that the user interface and editor in this package are among the best available for the ST. In many ways the GST shell uses the GEM environment better than any of the other packages. It is well designed and intuitive. You can compile, assemble and link a program from a single drop-down menu. If you have already compiled, you can just assemble and link. You may also link an assembled file. The program keeps a log of your activities so that if you are not paying attention during a compile or program execution, it tells you the results. It is a very well-done interface.

Atari
wants you to succeed in your product development.

GST uses a one-pass compiler that works reasonably quickly. The syntax for handling in-line assembly is a little different than the other compilers, but it▶

is not excessively cumbersome.

The compiler has an option to pause on locating an error condition. However, for some reason, when we used this, it seemed to hang for a while, then go on randomly after several presses of the Return key. One useful command option is the ability to have your comments from your C source code preserved and passed on to the assembler code.

The GST assembler is a very quick two-pass assembler. You have the option of producing either relocatable or nonrelocatable code. It is intelligent enough to spot long branching instructions that could be replaced with short branches and it warns you about them.

The GST linker is the same one used by Lattice C. It can be instructed to either include an entire library, or import only those routines required for linkage. In tight programming environments, this lets you optimize code size. The linker requires a command file to produce its output. The command file tells the linker what files to link and which libraries to search or include. According to drop-down menu options, the output and error messages of the compiler, assembler and linker can be re-routed to a file, a list, the console, or an auxilary device. Unfortunately, this feature and the command file for the linker are not well documented and we couldn't get the redirection to work properly.

A real problem we ran into was the size of the intermediate code. With two double-sided disks—one for the system and one for the source code—we often had to stop and delete files from the disk. On any given compile, five files are produced and retained. This situation makes a single-sided, single-density disk system useless for any significant development.

EDITOR

The GST editor is the highlight of the package. It is GEM based and very similar to 1ST Word. If you are switching back and forth between word processing and program editing, this saves the necessity of readapting to a different interface each time. The function keys have been redefined to suit actions appropriate to a programming environment—such as "move to the end of the line," etc. Like 1ST Word, the function-key boxes are drawn at the bottom of the screen and the programmer has the choice of keyboard input or mouse input. The editor is slow to respond, so it is easy to "get ahead of it" while using the cursor keys.

The ability to go to a specific line number is an important feature this editor lacks. GST avoids the necessity of line numbers by listing each function as it is compiled. It then lists the offending line. Usually this is enough to get you in the appropriate area of code if you are using the GST compiler with the **−m** option. Another missing feature is continously scrolling scroll arrows. GST has implemented the continuous scroll arrows in the dialog boxes, but for some reason they didn't use them in the editor itself.

In spite of these problems, this is the editor we use with large-scale code. It is easy to use and handles extra-large files easily and without problems. With just a little bit of clean up it would be worth the purchase price of the package alone.

SUPPORT

Antic Publishing, Inc. is supporting this product. They offer bug fixes free and upgrades at a nominal charge. As of June 1986, GST is planning a toll-free 800 number for technical assistance.

CONCLUSION

Those features of GST which have been implemented, have been implemented very well. The shell and editor use the GEM environment better than any package currently available. The fundamental difficulty of the GST C package is the features which were not implemented. As it stands, GST C is reminiscent of Small C from Dr. Dobbs. Because of its low price, we would like to be able to recommend it to beginning programmers. But without structures and other essentials of the C language, we are unable to do so. The tricks, and techniques that this package would force one to use might create bad programming habits. However, GST C has the potential to be a real winner if the company chooses to finish it. They are trying to do that as fast as they can. We look forward to seeing the finished product.

SUMMARY

Each of the development packages mentioned has certain advantages. If you are a software development house, or have a program ready to market, strongly consider the Developer's Toolkit with its CompuServe connection and marketing support. For a development compiler, Megamax C is, without question, the best available on the Atari. It will reduce your compile/test turn-around time by at least a factor of five. By taking advantage of the make utility and code modulation, this factor can increase to as much as ten. Unfortunately, at $200, it is a bit expensive for many people. With many students, and hobbyists, price is the single most important factor. The Lattice compiler, at $50 less, does not really offer a significant savings. However, its compatibility with its IBM cousin is a real plus. On the IBM PC, the same compiler, without an editor, has a list price of over $400. On the lower end, the GST package comes in with a very nice shell at half the price of the Lattice package. Depending upon how quickly GST upgrades it, this package is one to keep an eye on. ■

REFERENCE:

- *COMPUTER LANGUAGE Magazine*, 131 Townsend Street, San Francisco, CA 94107, (415) 957-9353

LIST OF MANUFACTURERS

Alcyon C (Developer's Toolkit)
Atari Corp., 1196 Borregas Avenue, Sunnyvale, CA 94086, (408) 745-2000
$300
CIRCLE 204 ON READER SERVICE CARD

Megamax C
Megamax, Box 851521, Richardson, TX 75085, (214) 987-4931
$199.95
CIRCLE 205 ON READER SERVICE CARD

Metacomco's Lattice C
The Catalog, 524 Second Street, San Francisco, CA 94107, (800) 443-0100 Ext. 133
$149.95
CIRCLE 206 ON READER SERVICE CARD

GST C
The Catalog, 524 Second Street, San Francisco, CA 94107, (800) 443-0100 Ext. 133
$79.95
CIRCLE 207 ON READER SERVICE CARD

ATARI ST C COMPILERS

	ALCYON	MEGAMAX	LATTICE	GST
Price	$300	$200	$150	$80
Full K&R	yes	yes	yes	no
Number af registers	8	6	8	0
Integer size	16	16	32	32
* Minimum disk size	711K	325K	538K	398K
**Minimum drives	2	1	2	2
GEM doc.	yes	yes	partial	list
Index	partial	yes	yes	no
ANSI extensians	partial	partial	full	no
GEM shell	no	yes	+ yes	yes

* Disk usage includes editors ond oll librory and link files.
** Minimum single-sided drives far *'practical'* usage.
\+ As af June 1986, Lattice is shipped with a GEM shell.

BENCHMARKS

DOODLE:	ALCYON	MEGAMAX	LATTICE	GST
Flappy campile	17:50	1:28	8:36	N/A
Floppy link	5:45	1:41	6:55	N/A
Hord disk compile	6:27	:40	5:15	N/A
Hord disk link	2:16	:48	2:10	N/A
End code size	16,705	15,854	36,154	N/A

DHAMPSTONE:	ALCYON	MEGAMAX	LATTICE	GST
Floppy campile	5:08	0:30	2:29	1:27
Flappy link	3:21	1:44	4:34	2:44
Hard disk campile	1:56	:12	:48	:45
Hord disk link	1:38	:26	1:22	:38
End cade size	18,154	11,075	19,000	25,652

DHAMPSTONE TIMES: (in seconds)	ALCYON	MEGAMAX	LATTICE	GST
Strings	7.87	7.60	7.57	10
Integers	2.03	2.05	3.11	4
Langs	.27	.25	.38	2
Unsinged	1.65	1.85	2.00	2
Double	.120	.72	.10	N/A
* Disk I/O	35.70	9.67	20.07	16

* Dhampstone creates a test file of 12,987 bytes by assigning a shart string buffer and writing it ta disk 1,000 times, ane byte at a time.

START SAVING NOW!

Atari 520ST 637.00

512 RAM, TOS on ROM, keyboard, 10 key pad, high res. monochrome SM124 monitor, SF354 DD, mouse, free $200 value software included.

520ST w/SC1224 RGB color monitor 768.00

16-BIT HARDWARE

Atari SF 354 DD/SS	149.00	SMM804 printer	189.50
Atari SF314 DD/DD	209.00	Avatex 1200 modem	88.00

8-BIT ATARI

130XE computer	125.00	XM801 printer	187.00
1050 DD	125.00	850 Interface	112.00
XM301 modem	35.00	Digital devices interfaces	CALL

SOFTWARE

Degas	25.00	Music Studio	39.00
Easy Draw	94.00	Pawn	29.00
FIS Strike Eagle	29.00	PC Intercom	78.00
Final Word	98.00	Silent Service	25.00
Leader Board	28.00	Spiderman	10.00
Mean 18	34.00	Sundog	29.00
Mud Pies	25.00	Time Bandits	29.00

1-219-233-5777

ELF SOFTWARE

P.O. BOX 6566 SOUTH BEND, IN 46660

Call or write to be put on mailing list. Order lines open 10 AM-6 PM EDT Mon-Fri. Prices reflect cash discount, MC/Visa add 4%. Shipping/handling: add 3%, $7 minimum. International orders may require additional 5& H charges. IN orders include 5% sales tax. All prices subject to change. All items subject to availability.

CIRCLE 021 ON READER SERVICE CARD

Megamax C

for the

Atari ST

Featuring

- One pass Compile • In-Line Assembly • Smart Linker
- Full Access to GEM routines • Register Variable Support • Position Independent Code • and much more...

System Includes:

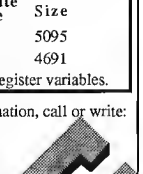
- Full K&R C Compiler (with common extensions)
- Linker • Librarian • Disassembler • C Specific Editor
- Code Improver • Documentation • Graphical Shell

Benchmark	Compile Time	Execute Time	Size
Sieve	70	2.28	5095
"Hello, world"	63	N/A	4691

*Times in seconds. Sieve with register variables.

$199.95 For more information, call or write:

Megamax, Inc
Box 851521
Richardson, TX 75085
(214) 987-4931
VISA, MC, COD ACCEPTED



CIRCLE 062 ON READER SERVICE CARD

PUBLIC NOTICE

GEMINI WILL MATCH ANY ADVERTISED PRICE & GIVE FREE SHIPPING

ORDERS SHIPPED WITHIN 24 HOURS
FULL EXCHANGE ON DEFECTIVES

Gemini Enterprises, one of Atari's largest dealers, will match any advertised price and ship FREE in continental U.S. Send certified check or money order, stating publication & page number of item you want price matching on, to —

GEMINI ENTERPRISES

692 Milford Road
East Stroudsburg, PA 18301

(717) 424-2248

$5.00 HANDLING CHARGE FOR APO & FPO ADDRESS.

CIRCLE 030 ON READER SERVICE CARD

ATARI ST USERS!*
ENTERTAINMENT JACKPOT
20 BIG PROGRAMS WRITTEN IN ST BASIC
72 PAGE MANUAL INCLUDED
ALL FOR ONLY $34.95

THE VISITDR
Text adventure with graphics. Your smart but odd companion must rendezvous with its mother ship

BDMB SQUAD
Text adventure with graphics. Find the terrorists' bombs in time

ADVENTURE CREATDR
Write your own adventure games. Extensive how-to-do-it instructions. A "framework program" is provided so you can fill in the details of your own games. The program is analyzed in detail. A powerful, fast machine language parser routine is provided and explained. BASIC graphics subroutines included

THERAPIST
"Talk to" your ST in natural English and it responds like a counselor. Similar to the famous ELIZA but "smarter"

MANSION
Text adventure. Find the second Mona Lisa.

3-D TIC TAC TDE
Challenging complexity

COLDR MDNITDR REDUIRED

(Disk and Manual).
Only $34.95 ($43.95 Canadian); M.O., VISA, MASTERCARD (include expiry).

*T M Atari Corp

THE WRITER
Watch your ST write poetry and prose, personalize the vocabulary and characters for party entertainment

CASINO
Lose your money at home. Includes Roulette, Blackjack, Craps, Cards — Faro, Baccarat, Draw Poker, Slot Machine, Wheel of Fortune, Keno

DTHELLD
Beat the computer

CHECKERS
A classic.

CRIBBAGE
The popular card game

BACKGAMMON
The ST is aggressive

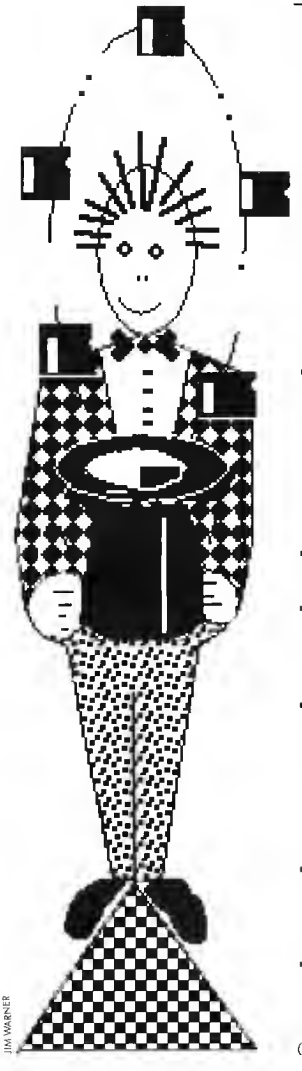
MENTAL
A great "psychic" illusion. The ST seems to possess amazing abilities

ANALYSIS
Convincing "personality analyses" — just for fun. Mimics such things as "color analysis" machines and explains how they work

CHARGE CARD ORDERS ONLY
Ph 800-628-2828 Ext 635

MARTIN CONSULTING
94 Macalastar Bay
Winnipeg, Manitoba
Canada R3T 2X5
(204) 269-3234

CIRCLE 061 ON READER SERVICE CARD



JIM WARNER

CLIPBOARD

- From the GEM Desktop, you can access an "**inactive**" **window** by first holding down the right mouse button, then, while holding it down, clicking the left mouse button on any item within the inactive window.
- A standard configuration 520 ST can safely handle a 200K **RAMdisk**
- If you need to **reboot** your ST, be wary of the Reset button. The Reset button only performs a "warmstart" which may leave unwanted data in the registers. To play it safe, it's best to use the power switch. Also, when rebooting a 1040 ST, turn the power off and *count to 15 before turning it back on.* It takes a while for the 1040 ST's power supplies to "zero out." This is particularly important when using a RAMdisk.
- To make a program **autoboot**. From the File drop-down menu, create a folder called AUTO. The program you wish to start automatically should then be dragged into the AUTO folder. The chosen program must have a .PRG extender. You can also use a .TOS program that does not make AES or VDI calls (since the AUTO folder is executed before GEM is initialized). Be sure to rename the file so that it has a .PRG filename extender. Now, power up your ST with the AUTO folder disk in drive A and your program will be executed. (See "The Amazing MouseTrap" in this issue for an example.)
- Instead of opening and closing windows to view files when changing disks, **press** [Esc]. This will cause the active drive to reread the disk.
- The ST only allows a maximum of six **desk accessories**. If you want to put more than that on your boot disk, but only want a select few to automatically boot into the Desk menu, change the extenders of those you don't want from .ACC to .ACX. The operating system will ignore the .ACX files.
- Though your program may be automatically relocated by GEM, there is a quick and dirty way for an assembly language program to find out where it is in memory. It makes use of the fact that a **jsr** pushes the PC (Program Counter) register onto the stack before jumping to the subroutine. The following code will load the absolute address of **LABEL** into address register 0 (A0):

```
        jsr LABEL              ;push the addr onto the stack
LABEL:  movea.l (SP)+,A0       ;now, pull it off
```

- When printing a text file to the screen from the Desktop and the **--MORE--** prompt appears at the bottom, you probably know that pressing any key will print another screen of text. Did you know that pressing [Return] will print only one line at a time? Try it.
- If you use the Developer's Kit, you can install the BATCH.TTP program as TOS-Takes-Parameters with a Document type of BAT, then anytime you open a .BAT (BATch) file, it will automatically run under the BATCH.TTP program.

*(**Editor's note:** Got an ST trick or tip to share? Send it to START, Attention CLIPBOARD.)*

MICROTYME

A DIVISION OF MICRO PERIPHERALS, INC.

P.O. BOX 368

KETTERING, OHIO 45409

ATARI® ATARI®

ATARI

520 ST's C'mon Now, Do It!	CALL
SF 314 Double Sided Drive	CALL
SHD 204 20 Megabyte Hard Disk	CALL
SC 1224 RGB Color Monitor	CALL
130 XE (8-bit Wonder of the World!)	125
65 XE	89
1050 Disk Drive	125
1020 Color Printer / Plotter	25
Power Supply 400/800/810 1050/850	15
Power Supply 600/800 XL, 130 XE	29
INDUS GT	219
Power Supply for Indus GT	15

PANASONIC

KX-P1080 5 NLQ MODES! NEW	219
KX-P1091 Rated the No. 1 Printer!	249
KX-P1092 80 col, True 180 cps	339
KX-P1592 136 col, True 180 cps	549
KX-P2131 L.Q. Daisy, 80 col	279
KX-P3151 L.Q. Daisy, 136 col	429
KX-P110 Ribbon, Blk	9
COLOR RIBBONS	11

CITIZEN

MSP-10	269
MSP-15	CALL

EPSON

LX-80 (80 col)	239
FX-85 (80 col)	379
FX-286 200 cps (135 col)	539

STAR MICRONICS

NX-10 (80 col) NEW MODEL	CALL
SG-10 (80 col)	229
SG-15 (135 col)	429
STAR SG-10 Ribbons	4

MODEMS

ATARI 1030	45
XM-301 Direct Connect	38
QMI 1200 ST (for 520 ST Complete!)	179
HAYES 1200 Smartmodem	399
US ROBOTICS COURIER 2400-100% Hayes!	429
PRENTIS P212ST 1200 bps, 100% Hayes!	239
SUPRA 1200 AT	179
SUPRA ST MODEM 1200 bps	179
VOLKSMODEM 1200	189
AVATEX Smart 1200 bps Special	99

INTERFACES/BUFFERS

ATARI 850 In Stock!	119
P.R. CONNECTION (100% 850 compatible)	66
CABLES - We've Got 'Em	CALL
U CALL (For Hayes, etc.)	39
U PRINT A	52
U PRINT A-64 with 64K Buffer	99
APE FACE XLP	59
SUPRA/MPP MICROPRINT	39
SUPRA/MPP MICROSTUFFER (64K Buffer)	69
SUPRA/MPP 1150	52

SOFTWARE and BOOKS

ST SOFTWARE TOO MUCH TO LIST	CALL
ALL titles from: Haba, VIP, Broderbund, Mark of the Unicorn, Hippo, Unison World, Migraph, Oss, Infocom, Atari, Michtron, SST Systems, Mirage Concepts, etc.	
We will have everything WORTH having!	
"THE C PROGRAMMING LANGUAGE" by B.W. Kernighan and D.M. Ritchie	19
8 BIT SOFTWARE FOR THE LATEST,	CALL
PAPERCLIP	39
PRINTSHOP	29
GRAPHICS LIBRARY #1, #2, or #3 (each)	16
O.S.S. BASIC XE	46
O.S.S. BASIC XL	36

MONITORS

TEKNIKA MJ-10 Composite Color	189
TEKNIKA MJ-22 RGB and Composite	279
THOMPSON Green W/Audio	85
THOMPSON Amber W/Audio	90
THOMPSON Composite Color	159

ACCESSORIES

ST COVERS, Heavy Grade Vinyl	8
ST MOUSE MAT, Matching ST Color	10
ST 6' Printer Cable	19
ST Modem Cable (To Hayes, etc.)	17
ST Monitor Stand, Swivel & Tilt	15
Disk File for 3.5" disks (holds 40)	9
Flip N File DATA CASE (holds 50)	8
Disk File, with Lock (holds 100!)	13
Rotary Disk File (holds 72)	15
Power Strip 6 outlet, (15 amp Surge)	15
Printer Stand, Heavy Duty, Sloping	13
ATARI "Standard" Joystick	6
6 Atari Serial I/O Cable	7
Compuserve Starter Kit	21
U.S. DOUBLER (Obl. Density for 1050)	49
"Duplicator"	129

PRINTER SUPPLIES

MAILING LABELS, White, 500 pack	3
per 1000	4
Blu, Pnk, Grn, Yel, 500 pack (200 ea)	9
per 500, any 1 color	5
per 1000, any 1 color	7
Big Labels, 1-7/16x4", White per 500	5
PRINTER PAPER, Micro-Fine perfs, 20 lb	
500 sheets, Pure White Bond	8
1000 sheets, same as above	14
Carton (2600 sheets), as above	29
PRINTSHOP "Rainbow" Color Paper Packs	
Pastels (5 colors) 50 sheets of ea	12
Matching Envelopes, 20 of each	6
Brights (8 colors), 50 sheets of ea	29
Matching Envelopes, 20 of each	10
ALL 13 colors, 50 sheets of each	39
Matching Envelopes, 20 of each	14
(Deduct 10% for 100/color paper packs)	

DISKETTES

Prices Are Per Box of 10 — **Minimum Order of 2 Boxes**

No. of	GENERIC		BONUS		WABASH	3.5" MICRO-FLOPPIES			
						SONY		VERBATIM	
Boxes	SS/DD	DS/DD	SS/DD	DS/DD	SS/DD	SS/DD	DS/DD	SS/DD	
2-5	8.50	10.50	10.50	13.50	10.50	20.50	29.50	15.50	
6-10	7.50	9.50	9.50	12.50	9.50	19.50	28.50	14.50	

Rainbow Colored Centech Disks (2 ea of 10 colors per pkg) 17

"Silver" Centech Disks (20 Pack) 17

TO ORDER, CALL TOLL FREE

1-800-255-5835

M-TH 9 am-9 pm • FRI 9 am-5 pm • SAT 10 am-2 pm EST

VISA

Ohio Residents, Order Status or Tech. Info Call (513) 294-6236

MasterCard

TERMS AND CONDITIONS

24 HR shipping on in store items • NO EXTRA CHARGES FOR CREDIT CARDS! • Minimum order $20 • C.O.D. to continental U.S. only, add $3 • Ohio residents add 6% sales tax • Please allow 2 weeks for personal or company checks to clear • Shipping/Handling: Hardware $4 minimum. Software and most accessories $3 minimum • Overnight shipment available at extra charge • We ship to Alaska, Hawaii, Puerto Rico (UPS Blue Label Only), APO, and FPO • Canadian orders actual shipping plus 5% minimum $5 • All defective products require a return authorization number to be accepted for repair or replacement • No Not Items or Returns • Due to changing market conditions, call toll free for latest prices and availability of product

CIRCLE 069 ON READER SERVICE CARD

520 ST

1 Megabyte

RAM UPGRADE BOARD

- *Fits under RF Shield*
- *Minimum Solder Connections*
- *Do it Yourself Installation*

$174.95

Dealer Inquiries Welcome

Send Check or Money Order to:

DIVERSE DATA PRODUCTS, INC.
1805 NE 164 Street
North Miami Beach, Florida 33162
(305) 940-0458
(305) 940-4763

Out of Country Orders add $10 Shipping
Use Int. Money Orders payable in U.S. Fund
Florida Residents add 5% tax

CIRCLE 018 ON READER SERVICE CARD

ST PROTECTION NOW AVAILABLE!

Disclone Service Won't Make You Nervous

Technical support, personal service, competitive prices.

Disclone full service quality tested diskette duplication, packaging, documentation production and processing ensures precise duplication, thorough quality control and expedient response to your requirements.

NOclone state of the art hardware based copy protection is true piracy protection for authorized allotments only. Each application is uniquely encrypted. Install routines are coded for nontransferrable hard disk allotments.

Disclone offers a choice of diskettes. Commitment dates are guaranteed. Fast turnover

- up to 1000 in 24 hours, any format.
- up to 10,000 in one week, any format.

disclone

DISCLONE SOFTWARE PRODUCTION SERVICES

1050 North Fifth Street, San Jose, California 95112

(408) 947-1161 **OUTSIDE CA:** 1-800-826-4296

CIRCLE 016 ON READER SERVICE CARD

Structured I/O

Complicated C Technique Simplified

BY HARRY KOONS

An indispensible C routine for the quick and elegant manipulation of disk files, uses GEMDOS and inherent C structures to save data as a contiguous block of memory, freeing the program from laborious file maintenance. Look in the STRUCTIO.STQ folder on your START disk for sample programs.

Many programs make extensive use of disk data files. A typical example is a database program which must save not only the user's data but also indexes, field and header information, and screen formats. This article describes a simple way to organize data files using structures in the C language.

THE PROBLEM

Designing reliable routines for disk I/O can be a frustrating experience for the professional programmer as well as the novice. The C library functions **sscanf( )** and **sprintf( )** can be used to read or write simple tables of data using a string containing conversion specifications. However, if you change the number or type of variables in the data file, the code may be difficult to maintain. The C language offers an elegant alternative for even the most complicated data. In C, structures provide a compact mechanism for organizing data on disk as well as in memory. ▸

Structured I/O...

Input and output are not part of the C language. However, some library functions akin to PRINT USING in BASIC are available for I/O, but using them to code routines that are reliable and easy to maintain is difficult. Kernighan and Ritchie (see *Reference* below) provide the following example. The call,

```
int i;
float x;
char name[50];
sscanf("%2d %f %*d %2s", &i, &x, name);
```

with input,

```
56789 0123 45a72
```

will assign 56 to **i**, 789.0 to **x**, skip over 0123, and place the string "45" in **name**. This is obviously an appalling situation. The programmer must keep track of the number of variables, the order of the variables, and the format of each variable—a formidable task for all but the simplest programs.

THE SOLUTION

Conceptually the data for one disk file can be collected in one contiguous block of memory. This block must then be written to or read from the disk. Graphics programs such as DEGAS and NEO handle the I/O for their pictures precisely this way. On the Atari ST, the screen uses 32,000 bytes which are located in a contiguous block of memory. One call to the GEMDOS functions **Fread( )** or **Fwrite( )** with the length and location of this block of memory is all that is needed to load or save a picture.

```
struct person {
    char name[NAMESIZE];
    char address[ADRSIZE];
    long zipcode;
    long ss_number;
    double salary;
    struct date birthdate;
    struct date hiredate;
 }employee ;
struct date {
int day;
int month;
int year;
int yearday;
char mon_name[4];
 };
```

FIGURE 1: A complicated C structure.

We can apply this idea to more complicated data as well. In C a structure is a collection of one or more variables, possibly of different types, grouped together under a single name for convenient handling. Not only are the variables grouped together conceptually in the structure, they are also physically grouped in contiguous bytes of memory. The C structure can thus serve for the image of our disk data file in memory.

Kernighan and Ritchie, in chapter six of their book, cite an employee payroll as an example of a complex data structure. This structure of type **person** (see *Figure 1*) contains within it two additional data structures of type **date**, as well as simple variables of various types. We know that the data are grouped together; all we have to do is find the location and the length of the structure in bytes in order to use the GEMDOS disk I/O functions.

We find the location by applying the address operator **&** to the stucture:

```
long location;
    •
    •
    •
location=&employee;
```

C provides an operator to determine the length of the structure. The expression **sizeof(object)** returns an integer equal to the size of the specified object. The size is given in bytes as if the object was type **char**. The object may be a variable, an array, or a C structure. We must declare **length** to be type **long** because a long is required by the GEMDOS functions **Fread( )** and **Fwrite( )**:

```
long length;
    •
    •
    •
length=sizeof(employee);
```

THE CODE

The program on your START disk, STRUCTIOTOS, is a test module which reads and writes a structure of type **person**. It is an example which illustrates several important programming techniques. In **main( )**, the structure is first initialized with zeros, and this is then verified by printing the elements to the screen. The variables in the structure are then set to sample values, and the result is again printed to the screen for verification. Following this, a file named EMPLOYEE.DAT is created and written to the disk using the structure values. The structure is again reset to zero, printed to the screen and, finally, it is refilled by reading the EMPLOYEE.DAT file then sent to the screen to verify a correct read.

The EMPLOYEE.DAT file is defined in the function **file_emp()**. Note that this function is completely symmetric: the data going in is the same as the data going out. The only argument is the mode, which is set to **RMODE_RD** for read or to **RMODE_WR** for write.

Notice also that the subroutine does not depend on the number of variables, the order of the variables, or the format of the variables within the structure. You can change the specification of the structure without making any change to the input/output routines!

The actual input/output is handled by the **do_io()** routine. This is also a completely symmetric and general routine that can handle all of the structured I/O for your program.

When I design a program, I assign each data file to a subroutine that looks like **file_emp()**. Several structures can be stored in one disk file like this:

```
length=sizeof(a);
location=&a;
result=do_io(handle, length, location, mode);
    •
    •
    •
length=sizeof(z);
location=&z;
result=do_io(handle, length, location, mode);
```

Only the name of the structure is changed in each three-line block of code. This routine reduces the chances of a coding error in the specification of the file structure because the structure definition determines the specification rather than the I/O routines.

Caution: The STRUCTIO program contains no error testing. Errors can occur at any GEMDOS function call. For example **Fopen()** will return an error if there is no disk in the drive and **Fwrite()** will fail if the disk is full. To avoid errors, place a formatted, non write-protected disk in drive A before running STRUCTIO.TOS.

The variable result, returned by **do_io()**, contains the number of bytes written or read. It can be compared with **length** to test for an error. Such errors could be tested in the **do_io()** routine if a general error message is appropriate or in the **file_emp()** routine if a message specific to that file is appropriate.

The techniques that I've outlined here can be the basis for all high-quality input/output. If you fill in this skeleton with thorough error testing, you will find yourself spending a lot less time designing and debugging your I/O routines. ■

REFERENCE:

- *The C Programming Language* by Brian W. Kernighan and Dennis M. Ritchie, Prentice-Hall, Englewood Cliffs, NJ

Mach1™ for the Atari ST

Mach1: multi-tasking Forth-83/GEM development system

With everything you need to develop stand-alone applications, including: integrated GEM editor, full GEM and DOS support, Motorola assembler, examples, and our 300 page manual.

Mach1 is interactive, so it allows you to experiment with the ST without going thru the compile-link-execute cycle. But when you do load in programs, look how we stack up:

Sieve	Compile	Link	Execute
Mach1	**00.7**	**00.0**	**4.41**
Megamax C	**70**	**24**	**3.83**
Hippo C	**58.4**	**1:37**	**8.4**

C's w/o register variables

(That's three times the execution speed of other Forth's) Note the turnaround time. It simply takes less time to develop your programs or finished products with Mach1.

all for only

$59.95

plus $5 S/H
CA Res. add 6.5%
VISA/MC COD

Palo Alto Shipping
PO Box 7430
Menlo Park, CA 94026
800/44-FORTH (Sales)
415/854-2749 (Dev. Support)

Available July 15, 1986

Original Macintosh version (v1.2) available for $49.95

CIRCLE 090 ON READER SERVICE CARD

TIMEKEEPER™ for Atari ST

$39.95

TIMEKEEPER™ is a plug-in battery-backed Real Time Clock Calendar for the ATARI ST Computers. The Timekeeper module plugs into the cartridge port on the ATARI 520/1040 ST Computer. A program is supplied with Timekeeper that operates as an accessory like the control panel. A feed through cartridge slot allows Timekeeper to remain installed while using other cartridges

The Timekeeper program automatically gets the time and date from the Timekeeper modules during the power up or boot sequence. The computer's own clock is set up and the Timekeeper is then disengaged until the computer is turned on again

AVAILABLE AT YOUR LOCAL ATARI DEALER,

or you may use your MasterCard or VISA to order direct by calling our **TOLL FREE NUMBER** below or send check or M.O. Please add $2.00 shipping or $10 shipping if outside U.S. California residents add 6% sales tax.

DEALER INQUIRIES WELCOME
800-654-2821 In California
800-624-6545 Nationwide

NAVARONE

CIRCLE 070 ON READER SERVICE CARD
NAVARONE INDUSTRIES, INC.
21109 LONGEWAY ROAD, SUITE C
SONORA CA 95370 • (209) 533-8349
TLX:WWI 650-230-9046

Lyco Computer Marketing & Consultants

LYCO COMPUTER MARKETING & CONSULTANTS

1080....$195

SAVE ON THESE IN STOCK PRINTERS

NX-10....Call

PANASONIC
1080.....195
1091.....225
3131.....249
1092.....309
1592.....419
1595.....599

LEGEND
808.....148
1080.....CALL
1380.....229
1385.....289

EPSON
LX80.....CALL
FX85.....CALL
DX10.....CALL
H180.....CALL
HS80.....CALL
FX286.....CALL
LQ800.....CALL
LQ1000.....CALL

SILVER REED
EXP 420 P.....CALL
EXP 600 P.....489
EXP 800 P.....649
EXP 770.....749

SEIKOSHA
SP-1000 A centronics.....175
BP-5200 I.....649
BP-1300.....469
Color Kit.....119
SP-1000 Ribbon...8.50

CITIZEN
120-D.....179
MSP-10.....285
MSP-15.....385
MSP-20.....325
MSP-25.....485
Premier 35.....469

OKIDATA
Okimate 10 XE....179
292.....CALL
293.....CALL
Okimate 20 ST....199
120 NLQ.....205
182.....214
192.....348
93.....CALL

JUKI
Juki 6100.....CALL
5510 Juki.....CALL
Juki 6300.....CALL
RS 232 Serial Bond.....55

STAR MICRONICS
LV12-10 (New)...CALL
NL-10.....CALL
NX-10.....CALL
NB-15.....CALL
SB-15.....CALL
SG-15.....367
SD-10.....319
SD-15.....438
SR-10.....469
SR-15.....578
SB-10.....589
Powertype.....297

MONITORS

ZENITH
ZVM 1220.....89
ZVM 1230.....89

TEKNIKA
MJ-10.....149

PANASONIC
TR-120 MDPA...CALL

HITACHI
MM-1218 12" Green.....99
CM-1406 13" Color W/Cable...179

MODEMS

Supra 300.....39.95
Supra 1200....149.95
XM 301.....39.95

INTERFACING

Microprint.....39.95
XETEC AT.....39.95
ATARI 850.....109
APE FACE.....39
U-PRINT A.....45

DRIVES

INDUS GT Atari....179
ATARI 1050.....129

FIREBIRD (520 ST)
The Pawn.....26.75
Star Glider.....28.75

ACTIVISION (520 ST)
Music Studio....34.75
Hacker.....26.75
Little People.....29.75
Paintworks.....40.75
Hacker II.....29.75

EPYX (520 ST)
Apshal Trilogy...24.75
Winter Games...24.75

SSI (520 ST)
Phantasie.....24.75
Baseball.....24.75

UNISON WORLD (520 ST)
Printmaster.....24.75
Art Gallery for II...18.75

TIMEWORKS (520 ST)
Word Writer.....CALL
Swift Calc.....CALL
Data Manager....CALL

ACCESS (520 ST)
Leader Board....24.75

SUBLOGIC (520 ST)
Flight Sim II...CALL
Set.....CALL

5¼" DISKETTES MAXELL
SSDD.....9.99
DSDD.....12.99

VERBATIM
SSDD.....9.50
DSDD.....12.99

BONUS
SSDD.....6.99
DSDD.....7.50

SKC
SSDD.....8.50
DSDD.....9.50

3.5" DISKETTES 3M
SSDD.....16.99
DSDD.....23.99

MAXELL
SSDD.....16.99
DSDD.....23.99

VERBATIM
SSDD.....16.99
DSDD.....24.99

SKC
SSDD.....14.99
DSDD.....19.99

NAME BRAND VIDEO TAPES from $4.49
Qty Discounts Available

SYSTEM PRICING AVAILABLE!

ATARI
1050.....129
SF314.....219
SF354.....175
130 XE.....CALL
65 XE.....CALL
520 ST Monochrome...CALL
520 ST Color.....CALL
SHD 204 20 Meg.....!
1040 ST.....CALL

BRODERBUND
Printshop.....28.75
Graphic Library I, II, III...18.75
Karateka.....19.75
Printshop Comp.....24.75

MICROLEAGUE
Baseball.....24.95
GM Disk.....24.95
Team Disk.....24.95
Stat Disk.....16.95

MICROPROSE
Silent Service....22.75
F-15.....22.75
Conflict in NAM.....24.75
Kennedy Approach.....22.75

UNISON WORLD
Printmaster.....24.75
Art Gallery.....18.75

ACCESS
Leader Board....24.75

SUBLOGIC
Flight Simulator...29.95
Night Mission Pinball.....18.95
Scenery Disks....Each 14.95

EPYX
Karate Champ...20.75
Apshal Trilogy...24.75

SSI
Wizards Crown...24.75
Gettysburg.....34.75
Gemstone Healer.....24.75
Phantasie.....24.75

ACTIVISION
Hacker.....15.75
Ghostbusters....15.75
Great AM Race...15.75
Music Studio....22.75

DISK DRIVE CLEANING KIT
$8.95
With Software
$17.95

"WE MAKE YOUR COMPUTER FUN TO USE!"

TOLL FREE 1-800-233-8760

In PA 717-494-1030
Customer Service 717-494-1670

or send order to
Lyco Computer
P.O. Box 5088
Jersey Shore, PA
17740

In stock items shipped within 24 hours of order. No deposit on C.O.D. orders. Free shipping on prepaid cash orders within the continental U.S. Volume discounts available. PA residents add sales tax. APO, FPO and international orders add $5.00 plus 3% for priority mail service. Advertised prices show 4% discount for cash, add 4% for MasterCard and Visa. Personal checks require 4 weeks clearance before shipping. We do not guarantee compatibility. We only ship factory fresh merchandise. Ask about UPS Blue and Red label shipping. All merchandise carried under manufacturer's warranty. Return restriction applicable. Return authorization required. All items subject to change without notice.

NEW HOURS!
Mon-Thur 9AM-5PM
Fri 9AM-6PM
Sat 10AM-5PM

NEW HOURS!
Mon-Thur 9AM-5PM
Fri 9AM-6PM
Sat 10AM-5PM

LYCO COMPUTER MARKETING & CONSULTANTS

STAccounts™
Integrated Accounting Package
For Atari ST Computers

No more spending thousands of dollars on an accounting package! For the price of some accounting packages alone, you can now purchase a full computer system that will satisfy your accounting needs and more. WHY!?

- STAccounts is fast, powerful, easy to use, and affordable.
- Fully integrated A/R, A/P, I/C, and G/L.
- All printed output/reports can be altered, amended, or entirely reconfigured to your specific needs utilizing the Report Generator.
- Easy as a click of a mouse you can change from one module to another.
- It's so easy to use.

The combination of the STAccounts and the Atari ST Personal Computer will provide you with the solution to all your needs at an affordable price.

VISIT YOUR ATARI DEALER TODAY AND FIND OUT FOR YOURSELF.

PARTIAL LIST OF FEATURES

Accounts Receivable (A/R) – Integrated with A/P, G/L & I/C.
* Account numbers and/or names * Customers' names and addresses * Invoicing/statements * Credit limit * 10 tax rates * Immediate account access * Backorders. REPORTS include Sales Turnover, Aged Receivables, Customer Statements and much more.

Accounts Payable (A/P) – Integrated with A/R, G/L and I/C.
* Account numbers and/or names * Suppliers' names & addresses * Remittance advices/statements * Immediate account access * Credit limits. REPORTS include Purchase Turnover, Aged Payables, an Supplier Mailing List.

Inventory Control (I/C) – Integrated with A/R, A/P and G/L.
* 10 product groups * Turnover by quantity and value * Alpha supported item lists * Inventory linked to A/P * Editing of invoice information. REPORTS include Inventory Turnover, Price Lists, Inventory Re-order list, and Stock Report.

General Ledger (G/L) – Integrated with A/R, A/P, and I/C.
*256 G/L Accounts * Immediate account access * Account details including budgets * Variance percentages for budget against actuals. G/L REPORTS include Profit and Loss Statement, Trial Balance, Balance Sheet and statement of accounts.

Help
STAccount provides you with on screen help through the use of TEDDY, a screen icon. If you become uncertain of your actions during the course of running the program, just click the left mouse button on TEDDY and a help screen for that particular section of the system will appear.

Report Generator
STAccounts Report Generator is a powerful tool integrated into the system. It will allow you to reconfigure reports using any of the system information. Provided with the system are a number of default reports which can be altered, amended, or entirely reconfigured for your individual company's needs, or create new reports as to your company requirements.

SYSTEM CONFIGURATION

STAccounts is designed to be used with either a 520 ST or 1040 ST, 0.5 or 1.0 megabyte drive with a monochrome monitor.

STAccounts and TEDDY are trademarks of Silicon Chip Limited
Atari ST is a trademark of Atari Corporation GEM is a trademark of Digital Research.

TO ORDER AND FOR MORE INFORMATION CALL

1-416-479-1880

U.S. ORDERS SHIPPED FROM U.S.
NON U.S. ORDERS SHIPPED FROM CANADA

VIP MARKETING
20 Steelcase Road, Unit 12
Markham, Ontario, Canada L3R 1B3

Send in this coupon with your credit card number, money order or check for U.S. $249.95.
In Ontario add 7% sales tax. No C.O.D.

☐ CHECK ☐ MONEY ORDER
☐ AMEX ☐ VISA ☐ MASTERCARD Expires _____

Account No. _____

Name _____

Company Name _____

Address _____

City _____ State _____ Zip _____

Phone _____

Signature _____

30 Day Money Back Guarantee: VIP MARKETING provides a 30 day money back guarantee that all claims and features listed in this ad are true.

STAccounts™
VIP MARKETING is the exclusive North American distributor.

CIRCLE 017 ON READER SERVICE CARD

Silicon Soothsaying

A Pro's Perspective

BY TIM OREN

Prophecy, prognostication and logical prefiguration. Tim Oren gazes into his digital crystal ball, scanning past computers to predict the future of the Atari ST and its line of software.

In 1963, the TX-2 was the hottest computer at MIT. One of the first machines devoted to a single user, it boasted a light pen and a vector display capable of showing up to 1,000 points at a time. It had less than 64 kilobytes of memory.

In 1968, an SDS-940 mainframe had 195 kilobytes of core memory, and shared its 96 megabytes of hard disk among 12 terminals, which were the first to be equipped with mice.

In 1973, the Xerox Alto had 128 kilobytes of memory, and one of the first bit-mapped displays. It stored 2.5 megabytes of data on a removable 15" cartridge disk, and was dedicated to one user.

In 1975, the Altair, the first hobbyist computer, was built by MITS. A company named Atari released a game called Pong. In the same year, Gordon Bell, a designer with Digital Equipment, predicted the birth of the Atari ST.

He didn't know its name, of course, but he graphed the memory size and cost of machines like these, and noticed that the price of hardware was dropping by a factor of four every three years. By his schedule, the Apple II appeared right on time in 1977. Since then, the rate of change has accelerated, resulting now in a one-megabyte machine for under $1000.

One result of this pattern, of course, is that you can buy more and more power for the same money. A 1040 ST with a hard disk has more raw power than any of the machines listed above.

Another interesting result is that we can predict the future of a series of computers by looking at what has happened with higher-priced machines. Similar predictions seem to hold for peripherals and programming as well. I am going to use this observation to do some uninhibited speculation and exhortation on what the future could and should hold for ST hardware and software.

Hardware trends are fairly easy to predict, though how they will eventually fit into Atari's product lines is anyone's guess! Let's start with multiple processors. The next logical step beyond one computer per user is many computers per user. Already, the many plug-in cards for PCs include MPUs such as Z80s or 6800s. Peripheral processors to handle I/O for mainframes have existed for decades, and in 1986 we are seeing the advent of massive parallel processing in supercomputers, with thousands of processors devoted to a single task.

The effect of this trend for the ST and other personal computers will depend on the availability of sophisticated CAD/CAM programs and fabrication facilities for custom VLSI chips (Very Large Scale Integrated chips). This means that the "extra" chips will be tailored for functions such as graphics or I/O bus control, for example, the Intel 82786, the Amiga's graphics chips, and the rapid appearance of EGA (Enhanced Graphics Adapter) clone chip sets. Atari's pre-announcement of a "blitter" chip for the next ST is an acknowledgement of the trend. Atari owns such VLSI design equipment, and has some of the best designers in the business.

The companion of multiprocessing is multitasking. (Multiprocessing devotes more than one chip to a program; multitasking lets more than one program run at a time.) Multitasking is already useful to the single user for print spooling and background compiles. In the future, it will become more important as the computer assumes the task of sifting through online systems and large databases for interesting material while the user continues to work.

Multitasking is supported on mid-range workstations such as Sun and Apollo, and is now appearing on single-user PC systems. While multitasking is largely a software problem, it requires some hardware support, in particular a memory management unit that keeps one errant task from destroying the others. The conflicting demands of multiple tasks also require hardware or software locking capability for peripherals and data files.

We can take it for granted that new machines will continue to have more memory. Four megabytes for a next generation ST seems easily in sight. Developers will, of course, quickly consume it and ask for more. The way out of the cycle is to introduce something that has existed for years on mainframes: virtual memory. If I am using a VAX and need 200,000 pointers to articles in a CD-ROM database, I can simply say:

```
long int art_locs[200000];
```

because it automatically handles the paging of my address space from RAM to disk. If I am using an ST, I must write an entire memory management scheme from scratch to accomplish the same task. The use of virtual memory implies a whole new layer of page-fault detection hardware, optimal caching logic, and so on. It also mandates the inclusion of a fast hard disk as part of the system.

PERIPHERALS

As the bandwidth of the basic microcomputer rises, the ability to transfer information to and from the user becomes limiting, and critically important. While using specialized chips to drive graphics and sound will help, other components are just as critical. (Bandwidth is the rate at which a processor or data channel can move information. Put simply, how fast can you push the bits?)

Any programmer who has complained about running out of screen space recognizes one of these limits. Space to display information is at a premium, but help is on the way. Upright, page-sized monochrome displays with 100 dot-per-inch resolution will be the graphics device of choice for the next generation.

Contrasting with the need for a tableau of information is the desire for a visually intense, natural presentation for color graphics. Experience with displays like the AT&T Targa indicates that the number of displayable colors is more important than resolution for such uses. Given this dichotomy, the ST's choices of monochrome or color modes can be seen as forward looking, not a retrogression. The either/or plug will have to go, though!

The most ruinous limitation in peripherals is the low bandwidth of user input devices. Conventional keyboards, joysticks, and mice have hit their limits, even given good user interface design. As evidence, I offer the continuing debate between the devotees of one-button and two-button mice. The subject of this religious warfare may be recognized as an example of "overloading," in which one symbol or action (point-click) is made to carry more than one meaning. Overloading is usually a sign of a severe design constraint, in this case, the limitation of the mouse to pointing in a two-coordinate plane. ▶

Soothsaying...

To break out of this bind, input devices must have more degrees of freedom. There seem to be two ways to get this. One is to take the keyboard approach, relying on the flexibility and trainability of the human mind to indicate its desires. Along this path lie MIDI keyboards for music input and pressure-sensitive touchpads with gesture recognition programs and software-definable control areas.

The second approach is to instrument or observe the human body, and let it behave naturally, moving in a virtual space defined by the computer's display (which might be viewed on a video projector). Researchers at MIT and the University of Connecticut have taken the tack of observing the movements with TV cameras, and extracting the relevant features. Limits in pattern-recognition methods, and the costs of such processing and the associated video gear, probably mean that it will be at least two generations before the computer will be watching us watching back.

For the short term, a significant breakthrough is the announcement of Jaron Lanier's "Glove," a device that is able to take independent input from five fingers as well as three-dimensional positioning. (See Infoworld, June 9, 1986.) Not only does this more than double the possible input bandwidth, it has profound implications for the design of virtual 3D workspaces. The continuing development of miniaturized strain gauges, accelerometers, and nonintrusive biomedical pickups for military uses says that the Glove is just a starting point.

The third trend in peripherals is the continuing merger of the computer into other forms of media, as correctly forseen by Alan Kay ten years ago. Sound and animation editors have progressed from toys on 8-bit machines to near professional quality in the current generation. The next generation will complete this trend, and begin adding hardware for integrating digital television into a total multimedia authoring and playback environment.

Already, computer power is being merged into a media standard as part of the Sony/Philips CD-I system. The ST's MIDI ports are an important toehold in this new world. Image scanners, frame grabbers, and genlock components will be needed next. Sorry, the oddball scan frequencies have to go, too.

Finally, the ability to access massive quantities of information will explode. CD-ROM for the ST, though much delayed, will become a reality, giving a desktop machine 200 times the data capacity of the Xerox Alto. It may also include the ability to retrieve sound tracks, computer graphics and video images.

The second part of this revolution, online communication and data retrieval, is well under way. Anyone who reads ANTIC ONLINE on CompuServe, who uses a computer bulletin board, or subscribes to Dow/Jones news service is already participating. A shakeout in this market is inevitable. Victory may go to the services that are best able to integrate their information with other sources such as CD, and with the microcomputer user interface.

An important question is how ST owners will get all of these goodies. Atari isn't large enough to develop them all. All this will be determined by the ST's fate in the marketplace.

The market for computer hardware and software add-ons is beginning to resemble that of "bolt-on" accessories for autos and aircraft. Given a wide variety of basic machines, a manufacturer will produce for those with the greatest distribution, with suitable mountings for accessories, and standardization across models.

The first "bolt-on" machine in microcomputers was the Apple II. It was followed by the IBM PC, and now the AT. Apple moved up to the Macintosh, but forgot to put in the bolt holes. They are now correcting this oversight. The true competition between the ST, the Mac, the Amiga and other computers is for acceptance as a "standard airframe," worthy of software and hardware investment.

SOFTWARE

My choice of example hatdware at the beginning of this article was not accidental. The systems described were used to produce:

- Sketchpad, the first graphics editor, written by Ivan Sutherland in 1963.
- NLS/Augment, the first "idea processor," developed by Doug Engelbart at SRI.
- Smalltalk, the windows-style interface, digital typefonts, Markup, and a host of other seminal applications by the Xerox PARC team.

Although our hardware today is superior to each of those systems, our software is markedly *inferior* to all of them, including one over two decades old! What has gone wrong here?

Part of the answer is quite natural. Minds that can produce such works do not stay long in the same place. They move on to new ideas that can be implemented on the ever-evolving hardware base. The transition of the ideas to the next lower class of hardware is left to other programmers, other companies. If their vision is less clear, if their resources are limited, the results may be a pale imitation of the original.

The exigencies of business promote this, too. As the price of the base hardware drops, the price obtainable for its software drops as well. Since the effort to implement the software stays largely constant, a company must spread its costs over a large number of units sold, or it will be very dead, very

quickly. This environment does not encourage risks on radical departures in software design, even if the concept has been proven on larger machines.

The same observation applies to hardware companies, since they have assumed the role of supplying operating systems and graphics software with their machines. The high prices of Apple computers are a direct reflection of that company's choice to invest in transfer and development of software technology. High development costs require high margins.

Atari is a different story, of course. When aiming for low price you do not invent—you rent. The innards of an ST are a roadmap of the company philosophy. Virtually every part is off the shelf, except for a few custom chips carefully optimized to replace the maximum number of components.

The software is the same. GEM and the rest were off the shelf from Digital Research. Atari wrote hardware drivers, made sure that some key applications were in place, and shot it out the door. To expect software leadership from Atari is unrealistic. Those who complain about lack of such support are wasting their time. It is simply inherent in the company's structure and position in a low-margin market.

This would all sound rather gloomy, but there is a way out. Outside developers, professional and amateur, can fill the need for good software, and be rewarded for doing so. To avoid the problems I've talked about, we have to keep a clear eye on two principles: leverage and idea transfer.

LEVERAGE

Leverage is making sure that your efforts pay off more than once. When dealing with software, leverage is spelled T-O-O-L-S. If you build the proper tools, you only pay that development cost once when you begin working a vein of good ideas. The rest comes free. Here are a few things I think we're going to need.

Tools come in many flavors: languages, libraries, utilities, and design and prototyping aids, to name a few. The basic languages for the ST are in place: C, Pascal, and assembler, as well as less known languages like Forth and Modula. What we need now are incremental compiling versions of these tools, with test runs conducted within the language environment. This is the way to break the interminable edit-compile-link-run-crash-debug cycle. Not only will this relieve frustration, but better applications will result from greater freedom to experiment.

Another big need is object-oriented extensions of the basic languages. ST implementations similar to the Macintosh's Object Pascal, the PC's Objective C, or Bell Labs' C++ would make it easier to implement good ideas borrowed from the Smalltalk world. The current ST models don't have quite enough power to run the most recent versions of Smalltalk itself, but this would be a good time to look at that language with an eye toward creating useful subsets, or transferring more features to other languages or applications.

Reusable source programs for GEM started with the DRI DOODLE program, which was more of a stock of plagiarizable code than a true library. Since then, a good deal of modular code has been made available through the CompuServe data libraries, due to contributions from Atari, DRI, and other developers, and as part of my own columns on ANTIC ONLINE. A logical next step would be to collect, codify, and extend this material and make it more generally available.

We also need a better skeleton application than the DOODLE program, which omits many important techniques. The best step would be in the direction of Apple's MacApp, which hides the basic event loops as part of system-level methods. ▶

STplus™

P.O. 1197 Berkeley, Ca. 94701

Save money through an authorized ATARI ST dealer! Fast delivery; knowledgeable support; & service. The ST is our specialty !!! Call for student, faculty, staff, or college discount.

30% off on all software!!

The Graphic Artist	$140	Thunder	$27.97
Personal Pascal	52.50	BTS, Paperclip	Call
DB Man, OneWrite	69.30	SunDog, Time	
Lattice C, Fortran	104.96	Bandits, Games	27.97 ea.
A-Calc	41.95	HippoWord	59.50

20% off all hardware!!

1040 ST Color	$959	Mono	$799
520 ST	$799		$639
20 MB Harddisk	$639	MacCartridge	$80
Brother 1109 NLQ	$239	Atari SMM	$175

Letter Quality Printers $259 and up

You can pay more, or you can pay less, but you can't buy better service!!!

Local *(415) 548-8999*
Calif. *(800) 874-4789*
Nat'l *(800) 433-6222*

$3.00 minimum shipping & handling

Amex • VISA • MasterCharge

CIRCLE 115 ON READER SERVICE CARD

Soothsaying...

Lacking the Object Pascal base on which to build such a system, a next best approach would be a standard top-level skeleton, with a source-code library of plug-in modules for such functions as drag recognition, window scrolling, and so on.

Linkers and debuggers are examples of tools I call "glue." A source-code-level debugger should be a part of any reasonably integrated development environment. Such a tool must also allow data to be examined in the context of its structural definition, not as unrelated bytes.

There are at least three good ideas in Sutherland's Sketchpad that haven't shown up on micros yet.

Leaving aside such long-range hopes, the single most useful bit of glue at this point would be a good incremental linker/librarian capable of handling both major ST object-module formats: Alcyon/Megamax and Lattice/GST. DRI's LINK68 is notoriously slow, and only handles one of these formats. A good implementation would produce a partially linked output, and then add any changed modules just before production of a loadable binary. That way, you only pay overhead for the code you have actually changed.

The Resource Construction Set is the first interface design and prototyping tool for the ST. There are a number of directions in which this notion could be improved and expanded. One would be adding real-time demonstration features, so that an entire interface could be shown in action, as well as created. Dan Bricklin's Demo Program is an example of this concept, executed in alpha mode on the IBM PC.

The current RCS objects and editing rules mirror the capability of the AES library code. Graphics-code libraries effectively extend the AES with new capabilities. The creators of such libraries could also build an enhanced RCS which incorporated new objects and rules to go along with the new functions. If the program was made extensible, using such methods as template objects and rule tables, end users could continue to augment the AES themselves.

IDEAS

In last issue's Perspectives, Dave Small quoted the programming axiom: "Steal from the best". I couldn't agree more. As you might guess, I find the pickings best when I look for ideas that have fallen between the cracks, or that haven't had time to appear on the latest machines. Another source often overlooked by "serious" designers is microcomputer games. There's not enough room here to do justice, but the following are some highlights that I think deserve attention. And remember what The Great Lobachevsky said. . ."Plagiarize!"

Graphics Editing and Visual Programming– There are at least three good ideas in Sutherland's Sketchpad that haven't shown up on micros yet. One is the notion of active handles on the graphic objects, allowing them to be "glued" together at specified points. For instance, a resistor symbol would have two external points of attachment. There was also the ability to combine multiple objects into a group, effectively defining by example a new class of objects with associated handles. Finally, Sketchpad allowed the entry of constraints such as parallel, coincident, and congruent to be automatically applied to the graphical elements. A more recent example of a constraint oriented graphics editor, with an improved user interface, is the Juno system developed at Xerox PARC.

Constraint-based editing is really an example of visual programming, in much the same way as PROLOG is a non-procedural way of implementing rule-based systems. Alan Borning's Thinglab program makes this aspect more explicit. There have been many attempts to implement procedural visual programming, with wide variations of success. One of the most interesting results is a game for the Macintosh called Chipwits, which uses a visual language to program a simulated robot. For more pointers to visual progamming ideas, see Myers' article on this topic, listed below.

New Ways to Think– Engelbart's NLS system was the earliest and maybe the best implementation of a hierarchically structured "idea processor." It incorporated several features we still lack, such as the ability to work cooperatively, and mix video images of the participants with the computer output. For those who like a free-form tool, Ted Nelson's Hypertext ideas of linked dynamic documents are attractive. At KnowledgeSet, we have been building a Hypertext system based on static documents stored on CD-ROM. The work going on at Brown University is even more exciting because it

combines several information sources in a dynamic hypermedia framework. One of the biggest upcoming challenges is finding meaningful ways to navigate in such massive databases

Microworlds—Smalltalk is still the best platform for making microcosmic simulations because it provides explicit mechanisms for creating types of objects and defining the ways they interact. Since Smalltalk has not been practical on a micro, the most interesting microworlds have been created by game designers. Bill Budge's Pinball Construction Set was the inspiration for the Resource Construction Set, and is superior in that it animates the results of the construction. The construction set metaphor has proven very popular on the Macintosh, being put to such diverse uses as animation, music, and calculators. A recent extension is the association of "field rules" with areas on the display, causing objects placed into an area to conform to predefined constraints.

When aiming

for low price you do not invent

— you rent.

Going Media— We are just now getting decent ST programs for editing text, static graphics and MIDI sound. The Audiolite presentations, available by using The Music Studio and Paint Works from Activision, give some idea of the results possible when these capabilities are mixed in a limited way. One of the hottest possibilities with the ST is to create an editing and playback system which integrates all of these data types and adds animated graphics, or eventually, images from a video source

ENVOI

The ST is a machine with an incredible amount of potential. If you have been feeling cramped by your old 8-bit machines, kick back and enjoy your new freedom. There's no need for individual developers to rebuild the same old programs; there are plenty of commercial organizations who will take the safe road and do just that. It's time to raise our sights, and realize how far we can go with some new ideas and a little careful engineering. ■

REFERENCE:

- *Sketchpad: A Man-Machine Graphical Communication System*, Ivan Sutherland, Proceedings Spring Joint Computer Conference, 1963, pp. 329-346
- *A Research Center for Augmenting Human Intellect*, Douglas C. Engelbart and William K. English, Proceedings Fall Joint Computer Conference, 1968, pp. 395-410.
- *Personal Distributed Computing: The Alto and Ethernet Hardware*, Charles P. Thacker, Proceedings ACM Conference on the History of Personal Workstations, Palo Alto, California, Jan. 9-10, 1986, pp. 87-100
- *Personal Distributed Computing. The Alto and Ethernet Software*, Butler W. Lampson, Proceedings ACM Conference on the History of Personal Workstations, Palo Alto, California, Jan. 9-10, 1986, pp. 101-131.
- *Toward a History of Personal Workstations*, C. Gorden Bell, Proceedings ACM Conference on the History of Personal Workstations, Palo Alto, California, Jan. 9-10, 1986, pp. 1-17.
- *A Study in Two-Handed Input*, William Buxton and Brad A. Myers, Proceedings *CHI* '86, Boston, Massachusetts, April 13-17, 1986, pp. 321-326.
- *Put That There: Voice and Gesture at the Graphic Interface*, Richard A. Bolt, Proceedings SIGGRAPH 1980, pp. 262-270.
- *Artifical Realty*, Myron W. Krueger, Addison-Wesley Publishing Co., Menlo Park, CA, 1983
- *Personal Dynamic Media*, Alan Kay and Adele Goldberg, IEEE Computer, March, 1977.
- *Smalltalk-80: The Language and its Implementation*, Adele Goldberg and David Robson, Addison-Wesley Publishing Co., Menlo Park, CA, 1983. ▸

My thanks to the organizers of the ACM History of Personal Workstations conference, which was the direct inspiration for this article.—*Tim Oren*

Soothsaying...

- *The Programming Language Aspects of Thinglab, a Constraint-Oriented Simulation Laboratory*, Alan Borning, ACM Transactions on Programming Languages and Systems, October, 1981, pp. 353-387.
- *Visual Programming, Programming by Example, and Program Visualization: A Taxonomy*, Brad A. Myers, Proceedings CHI '86, Boston, Massachusetts, April 13-17, 1986, pp. 59-66.
- *Dream Machines*, Ted Nelson, Hugo's Book Source, Chicago, IL, 1974.
- *A New Home for the Mind*, Ted Nelson, Datamation, March, 1982, pp. 169-180.
- *Reading and Writing the Electronic Book*, Nicole Yankelovich, Norman Meyrowitz, and Adries van Dam, IEEE Computer, October, 1985, pp. 15-30.
- *An Overview of Information Retrieval Subjects*, Martin Bartschi, IEEE Computer, May, 1985, pp. 67-84
- *Generalized Fisheye Views*, George W. Furnas, Proceedings CHI' 86, Boston, Massachusetts, April 13-17, 1986, pp. 16-23

The Music Studio
Paint Works
Activision
P.O. Box 7287
Mountain View, CA 94039
(415) 960-0410

You need . . .
"function_aid"

- Stop fumbling for your manuals
- Put your Macros in front of you
- Rests on top of your *ST
- Adjust window to suit your light
- When finished, remove & cover keys
- 4 Blank inserts included (2" x 12")
- Designed & made only for the *ST

*ST is a trademark of Atari Corporation

★ Indispensible for business programs (word processors, telecommunications and data bases) etc.
★ Just about any program you use needs · function_aid
★ If you own an Atari *ST you definitely need this!
★ Unconditional 5 year Guarantee
If you break function_aid return it and we will replace it at no charge to you.

STATIC ENGINEERING, INC.
P.O. Box 570, Bristol, CT 06010
Phones Open (9 a.m. - 9 p.m. Eastern) (203) 879-4671

Please send _____ function_aid (s) at our Special Introductory Price of $19.95 (reg $24.95) each ..... _____
Add a shipping charge of 1.75 each ..... _____
(CT residents add sales tax of 1.60 each) ..... _____
TOTAL _____

Name _____
Address _____
City _____ State _____ Zip _____

☐ Master Card ☐ VISA ☐ Money Order

Card No. _____ Expiration Date _____

(Allow three weeks for delivery)

SHIPPING OCTOBER!

CIRCLE 114 ON READER SERVICE CARD

WORD FOR WORD™ ST

A crossword game for the ATARI ST!

It's a challenging new game in which the players take turns creating words on a playing board. Here's what reviewers have to say:

"...the whole game design is extremely user-friendly...a winner." ANALOG COMPUTING June 1986

"...It's easy to use the mouse to design and save your own board layout...makes the game even more fun." ANTIC, April 1986

"...I am very impressed with Word for Word...full utilization of GEM...solid performance...a joy to play...attention to detail...excellent product." ST APPLICATIONS Jan-Feb 1986

To Order

Contact your Atari ST dealer or send $39.95 plus $3.50 for shipping and handling ($43.45) California residents add $2.40 sales tax ($45.85)

MasterCard or Visa accepted

Bay View Software
177 Webster St., Suite A-295
Monterey, CA 93940
(408) 373-4011

Works with color (medium resolution) or monochrome monitor

CIRCLE 007 ON READER SERVICE CARD

METACOMCO

The quality source for 68000 software
Announces

CAMBRIDGE LISP

THE SYMBOLIC LANGUAGE FOR ATARI ST

An interpreter/compiler providing a complete LISP development environment for $199.95

also available

Lattice 'C'	MCC Pascal	Macro Assembler
$149.95	$99.95	$79.95
APL-Symbolic	**APL-Keyword**	**Menu+**
$245.00	$149.95	$29.95

Contact your local dealer or call:
TEL: (US) 800-AKA-META (CAL) 800-GET-META
EUR (UK) 44-272-428781
TELEX (UK) 444874 METACO G
Add 6½% tax if CA resident

Metacomco CIRCLE 066 ON READER SERVICE CARD
5353 #E Scotts Valley Dr., Scotts Valley, CA 95066

Lattice is a registered trademark of Lattice Inc
Atari ST is a registered trademark of Atari, Inc

Software Discounters of America

S.D of A

Orders Outside PA—1-800-225-7638
PA Orders — 1-800-223-7784
Customer Service 412-361-5291
Open Saturday

- **Free Shipping on orders over $100 in continental USA**
- **No surcharge for VISA/MasterCard**
- **Your card is not charged until we ship**

Your ST Software Connection!

ABACUS BOOKS	
ST Gem Prog Ref	
ST Graphics & Sound	
ST Internals	
ST Logo	
ST Machine Language	
ST Peeks & Pokes	
ST Tricks & Tips	
Call For Low Prices!	
*Optional Program Disks are available for all books!	Call
ABACUS SOFTWARE	
Assem Pro	CALL
FortrHMT	FOR
ST File Pro	LOW
ST Test Pro	PRICES
Text Designer	
ACADEMY	
Typing Tutor	$23
ACCESS	
Leader Board Golf	$25
ACCOLADE	
Mean 18 Golf	$29
ACTIVISION	
Borrowed Time	$33
Hacker	$29
Hacker 2	$33
Little Computer People	$33
Mindshadow	$33
Music Studio	$39
Paint Works	$44
ARTWORX	
Bridge	$19
Compubridge	$19
Mail List	$14
Strip Poker	$25
BATTERIES INCLUDED	
D.E.G.A.S.	$25
D.E.G.A.S. Elite	$49
Home Pak	$33
I*S Talk	$49
Issur Portfolio System	$129
Paperclip Elite	Call
Thunder	$25
Time Link	$33
CENTRAL POINT	
Copy 2	$25
ELECTRONIC ARTS	
Financial Cookbook	$33
EPYX	
Championship Wrestling	Call
Rogue	$25
Temple Apshai Trilogy	$25
Winter Games	$25
World Games	Call
FIREBIRD	
Starglider	$29
The Pawn	$29
FTL/SOFTWARE HEAVEN	
Sundog	$25

INFOCOM	
A Mind Forever Voyaging	$29
Ballyhoo	$25
Cutthroats	$25
Deadline	$33
Enchanter	$25
Hitchhikers Guide to the Galaxy	$25
Infidel	$29
Seastalker	$25
Spellbreaker	$33
Trinity	$25
Wishbringer	$25
Witness	$25
Zork 1	$25
Zork 2 or 3	$29
MARK OF THE UNICORN	
Hex	$25
PC Intercomm	$79
Final Word	$79
MEGASOFT	
ST Copy	$23
X Rated Graphics Library Disk for Printmaster	$16
MICHTRON	
Bulletin Board System	$49
Calendar	$19
DOS Shell	$25
Kissed	$25
M-Disk	$25
Major Motion	$25
Mighty Mail	$33
Mi-Term	$33
Personal Money Mgr.	$33
Pro-Football Wizard	$25
Soft Spool	$25
Time Bandits	$25
Utilities	$39
MICROPROSE	
Silent Service	$26
MINDSCAPE	
Bratacuss	$33
Defender of the Crown	Call
King of Chicago	Call
Sinbard: Throne of the Falcon	Call
OSS	
Personal Pascal	$49
Personal Prologue	Call
OMNITREND	
Universe 2	$49
ORIGIN	
Ultima 3	$39
PENGUIN/POLARWARE	
Crimson Crown	$14
Oo-Topos	$14
Transylvania	$14
QUICK VIEW	
Zoom Racks	$49
REGENT	
Regent Base	$65

SIERRA ON LINE	
Black Cauldron	$25
Hint Books	Call
Kings Quest 2	$33
Ultima 2	$39
Winnie the Pooh	$19
SSI	
Phantasie	$25
SUBLOGIC	
Flight Simulator	Call
Jet	Call
SYNAPSE	
Breakers	$29
Mindwheel	$29
TELARIUM	
Amazon	$33
Fahrenheit 451	$33
Nine Princes in Amber	$33
Perry Mason Case of the Mandarin Murder	$33
TIMEWORKS	
Data Manager	$59
Swiftcalc	$59
Sylvia Porter's Personal Financial Planner	$59
Word Writer	$59
UNISON WORLD	
Art Gallery 1	$19
Art Gallery 2	$19
Print Master	$26
VIP TECHNOLOGIES	
VIP Professional	$99
VIP Professional Lite	$65
XLENT	
Rubber Stamp	$25
ST Music Box	$33
Typesetter	$25
ACCESSORIES	
Bulk Disks 3½	Call
Compuserve Starter Kit	$19
Disk Case (Holds 30)	$9
Dow Jones Starter Kit (5 hr)	$14
Dust Covers	Call
Kraft Joystick	$9
Panasonic 1080 Cps Dot Matrix Printer	Great Deal
Panasonic 3131 Letter Quality Daisy Wheel Printer	Call
Supra 20 Meg Hard Disk Drive	Call
Supra 300 Baud Modem w/Omega Terminal	$59
Supra 300/1200 Baud Modem w/Omega Terminal	$139
Wico Boss	$12
Wico Bat Handle	$17

Hours: Eastern Time
Mon.-Fri. 9AM-5PM
Sat. 10AM-5PM

P.O. BOX 111327—DEPT. ST—BLAWNOX, PA 15238

***Please Read The Following Ordering Terms & Conditions Carefully Before Placing Your Order:** Orders with cashiers check or money order shipped immediately. Personal & Company checks allow 3 weeks clearance. NO C.O.D.'s! **Shipping: Continental U.S.A.**—Orders under $100 add $3, **free shipping on orders over $100. AK, HI, FPO, APO—add $5 on all orders. Canada & Puerto Rico—add $10 on all orders.** Sorry, no other international orders accepted! **PA residents add 6% sales tax on the total amount of order including shipping charges!** Prices subject to change without notice. **REASONS FOR CALLING CUSTOMER SERVICE—412-361-5291 (1)** Status of order or back order **(2)** If any merchandise purchased within 60 days from S.D. of A. is defective, please call for a return authorization number. Defective merchandise will be replaced with the same merchandise only! **NO CREDITS!** After 60 days please refer to the manufacturers warranty included with the merchandise & return directly to the manufacturer. **Customar service will nol accept collacf calls or calls on S.D. ol A.'s 800# order lines!**

CIRCLE 112 ON READER SERVICE CARD

Transportable GEM

IBM to ST

BY MARK SKAPINKER

When Digital Research first introduced GEM, the big drawing card was the ease with which developers could port graphic-oriented software between different processors. Just how easy is it? Mark Skapinker of Batteries Included describes his experiences developing "trans-compatible" software for the IBM PC and the Atari ST.

When Batteries Included decided, in March 1985, to develop the ISGUR Portfolio System for MS-DOS using a graphics interface, GEM was the only major graphics system available for the IBM PC. Since we also knew it would become the interface on Atari's new ST computer, GEM was the natural choice for our software development.

THE TEAM

We established an in-house GEM development team in our Toronto offices, consisting of myself; Chris Bailey and Steve Couchman, our software engineers; Keith Hope, hardware engineer; Lloyd Speyer, tester; and Mike Reichmann, president of BI (with a special love for software development). Since the ST was not then available (and would have inferior development tools when it did become available), we decided to create our software on the IBM and port later to the ST. We have since developed—and are continuing to develop—a number of ST GEM applications using the MS-DOS / IBM PC environment as a host machine.

Chris Bailey, a hacker at heart, volunteered to dive into the depths of GEM and become the "GEM resource person" for both us and our outside developers. To accomplish this, he undertook two projects. First, since some people don't like mice, we wanted our GEM programs to be able to activate drop-down menus via the keyboard. By pressing a given function key, you would get a menu dropdown to fall, and by pressing the first letter of an item description, you could activate that element. Secondly, we needed a tool that would let us to use the GEM play and record functions (**appl_record** and **appl_play**) in a more usable and efficient manner than was provided with the GEM toolkit. As is typical with programming, these two tasks proved much more complicated than expected.

We discovered that once a menu item has been selected, control of the mouse is handed over to GEM exclusively, so how could we then see which key was pressed? Our solution was to have a desk accessory running that would look for keys at that time (since desk accessories could run as a separate task at the same time as a program). Thus, the desk accessory would intercept keystrokes from the keyboard at the time that the dropdown was lowered. Elegant, but complicated to say the least!

appl_record and **appl_play** should allow you to "record" mouse and keyboard movements then "play" them back at a later stage. Chris found these functions (particularly **appl_record**) very inefficient. For example, it might take 500 bytes of storage space to record a mouse movement across the page. Add mouse positions and timers and it becomes very large. He decided to write his own, more efficient version of these functions, which we called **replays**. Eventually, both **replays** and the function key accessory were combined into one desk accessory.

While Chris explored GEM, I began developing the ISGUR Portfolio System and was able to devote my time to the application side of the project. Steve Couchman joined us as our communications expert and began developing 1*S-Talk under MS-DOS GEM. He also agreed that as soon as the ST was available to us, he would become our ST expert.

In the meantime, Keith Hope performed magic on our MS-DOS machines. He added a 80286 board with two megabytes of RAM disks, cache memory, and a 40 megabyte hard disk to my Compaq. The Intel 80286 chip is the same processor found in the IBM PC AT, only this one ran at nine MHZ instead of six. The cache memory allowed the original 8088 chip in the Compaq to efficiently determine whether to access the disk for data, or whether the data was in memory already, thus cutting down on disk I/O. This effectively gave me a machine running 10 to 20 times faster than a regular IBM PC and gave us the freedom to concentrate on developing software. Truly a developers' heaven. We have become quite spoiled from all these hardware tools, as well as the myriad of MS-DOS software tools (Lattice C, Phoenix's linkers, debuggers, Lints, etc.) which are very sophisticated, bug free, fast and reliable.

EARLY ST

When the ST arrived later in the year, we knew we should start developing for it as soon as possible. But we were faced with a machine with almost no software tools, let alone fast hardware peripherals like hard disks. Pioneer ST developers will remember those amusing early version numbers: C compiler version 0.F, linker serial number 0000-AAAA. The time was ripe for system software to be developed on the ST. We knew that if the machine was to survive, it would need some useful application software reasonably quickly. We decided to immediately try our hand at porting ISGUR Portfolio System and 1*S Talk. Steve spent two weeks with the inadequate tools provided with the early ST—overflowing his floppy disk space, crashing. He later realized that he accomplished about as much in those two weeks as he now achieves in two hours.

After some frantic calls and long meetings, we decided to take a different approach: We would become GEM experts and concentrate on developing superior GEM applications under MS-DOS. At the proper time, we could port our code to the ST and make the necessary changes to take full advantage of the Atari hardware. In the meantime, Steve became a beta tester for everything we could find on the ST. He spent countless hours trying every compiler, linker, editor and got to know every available piece of systems software for the ST. The creative development environment was becoming a frustrating environment.

To make the IBM-ST port as straightforward as possible, the decision was made to develop our software in the C language. C is well known as an ideal language for cross development purposes. We opted for the Lattice compiler on the IBM because it was well suited to GEM, we had experience with it, it supported full C, and it was a mature system. Also, the developers at DRI had used Lattice to develop the GEM bindings and therefore recommended it. We also wrote our code in a structured, modular fashion—all data definition and machine specific codes were external, all I/O was localized in one position. Using C, we were able to write code that referred to external functions—such as memory allocation and I/O—in a very logical manner. (We found the header files PORTAB.H and MACHINE.H to be very useful in this regard.) We were determined to write code that would behave itself under ▸

Transportable GEM...

GEM; it is very easy to write non-standard code. We wanted to behave first—then, at a later stage, fine tune the code for speed.

We were fortunate to be provided with an excellent role model. BI's only ST software at that time was DEGAS (an outside acquisition from Tom Hudson). Its sales were better than anyone ever imagined. Within the first three months, we sold more copies of DEGAS than we expected to sell in the first year. We also found our international sales very significant. Other developers seemed to be finding ST sales to be a bit tough. This proved to us there was a strong market for solid, well written ST products, and that it was very worthwhile to develop other powerful ST programs.

After fall Comdex, we decided that the ST's time had come. We elected to port the ISGUR Portfolio and I*S Talk over to the ST.

ST TOOLS

There are now some good hard disks for the ST and some usable development software has started appearing. The situation is improving rapidly, but some of our biggest frustrations are rooted in not knowing what to blame for a bug. When we were porting the ISGUR Portfolio System from MS-DOS, we had almost one megabyte of source code to transfer! We decided to use the Megamax C compiler on the ST side. It seemed a solid, fast and quite efficient compiler.

Megamax compiled our code very quickly, but the system sometimes crashed during the link. Was our code too large?

We decided to create our software on the IBM and port later to the ST.

Were we passing some limits? Eventually we and Megamax discovered there was a problem in the memory allocation of the linker. We used this frustrating opportunity to fine tune our code and manually perform a lint. (Lint is a program that examines C code for correct syntax and structure. Under MS-DOS we could simply have used such a program. The closest we could get on the ST was to manually go through the code line by line.) Megamax had presumed that TOS would allocate memory as documented; this was not the case. (With TOS, each time you request memory, you must free it. Otherwise, after a certain number of memory requests without "free-ups," you will bomb. One solution is to determine in advance your total memory needs and request it all at once.)

The port for our two programs was fairly straightforward—which is not to say that it was completely simple. Under MS-DOS, one can choose which "model" to develop under. This is not true for the 68000 machines. A model in the 8086 processor indicates how large program and data address space should be (and thus the compiler creates two or four-byte pointers). The 68000 only uses four-byte pointers. Structures not byte aligned under MS-DOS had to be reconsidered. Structures were forced on to positive bytes, thus changing their sizes. But thanks to the portability of C, the incredible similarities between MS-DOS and GEMDOS, and the similarities of GEM in the two environments, it was not too bad. Where no GEMDOS call existed, a close approximation could always be found. On the other hand, MS-DOS is far more forgiving if you ignore directory paths in I/O operations. (For example, MS-DOS will accept A:FILE.DOC, while GEMDOS insists on A:\FILE.DOC).

THE SIX WEEK PROGRAM

In March 1986 my wife and nine-month-old daughter went to Israel to spend six weeks with my in-laws. For myself, I saw the opportunity of six weeks in which to develop an ST product. Having come from the MS-DOS world, I knew just how successful add-on memory resident tools—such as Borland's Sidekick, and Lightning—were on the IBM PC. BI had already created BatteryPak, a set of useful desk accessories for the Macintosh which was doing well. With GEM, we also had the desk accessory capability available. I am a terrible speller and find a spelling checker vital to my work. I did not want to develop "just another spelling checker," so I began planning the ideal writer's tool for the ST. My only restraint was *time*. I needed to finish the bulk of the project in six weeks, and was determined to be completely finished before the 1986 Summer CES.

I also had some hopes about the effect of this product on the ST world. My responsibilities at BI include looking at the marketplace, trying to understand where things are going, and trying to create a market environment that will help such machines as the ST to succeed in a big way. I have looked at all the software coming out for the ST and am convinced that the only way the ST will succeed is if a consistent interface is used for all of its programs. What will revitalize the home

computer marketplace is a common interface like that found on other consumer products such as cars, stoves and TVs. We have the GEM interface and, for better or for worse, we should make it work for us all. I hoped products like Thunder! would motivate other ST developers to follow the rules in developing well-behaved GEM applications. Accordingly, I decided to develop a program that would work with *any well behaved GEM ST product*. I just hope that we are not all hurt by some of the early applications coming out for the ST– it is not good enough to ship piles of software if they are not of decent quality. Some early GEM programs, for example, leave "holes" in the screen after desk accessories have been activated. Ultimately, such programs will not be worthwhile and financially they will fail. Enough preaching.

I decided the only way I could develop the system in six weeks was by developing a bug-free program on my Compaq under MS-DOS GEM, and only then porting it over. The big lesson I had learned from our first ports was to develop just the "core" under MS-DOS, test it thoroughly, port it to the ST, then develop that last ten percent of the project (which always seems to take fifty percent of the time) on both machines simultaneously. This way, I would know the code worked *before* it got to the ST. The core of the program for me was a working, efficient version of the program that lived well in the GEM world. Any bugs would be easy to identify and fix. Once on the Atari, I could spend time making the code ST'ish.

THUNDER!

Thunder! is made up of two programs: a real-time desk accessory which we initially called Thunder, and a stand-alone program, which we were calling Bolt at that time (Thunderbolt). The user interface is written mostly in C, with the keyboard interrupts and the dictionary lookups written in assembler. The dictionary consists of 50,000 words intricately squeezed into 80K. These are 50,000 real words; thus walk, walks, walked and walking are considered *one word*, effectively giving many more than 50,000 words.

The desk accessory part of the program looks up words in the dictionary as they are typed in. The program thus intercepts keys from the keyboard as they are typed in. These keyboard interrupts were reasonably close to the way we had previously handled our function keys accessory, so we already had the basic routines in 8086 and 68000 assembler. We had the dictionary routines in both assemblers as well. Thus, we were able to develop the C routines independently of the assembler routines.

On the 8086 side, we used Lattice C small model (data and program less than 64k) to create efficient code. We were determined to maintain only one set of source code; the same program would be maintained for both machines. We could do this very smoothly by using the MACHINE.H and PORTAB.H files supplied by DRI. MACHINE.H is a header file written by Digital Research that defines machine specifics about routines used by some GEM routines, such as defining of the size of ADDR pointers. PORTAB.H defines variables

This effectively gave me a machine running 10 to 20 times faster than a regular IBM PC . . . Truly a developers' heaven.

such as BYTE, WORD, BOOLEAN, and LONG which were used instead of char, int etc. By using these header files, our code became very portable between the two machines. Where specific code was needed for a particular machine, we simply defined PCDOS or MC68K in the header files and used, **#ifdef** statements.

An example of the **#ifdef** statement in using the OSBIND.H file would be:

```
#if MC68K #include "osbind.h" #endif
```

or even defining the name of the main module:

```
#if MC68K main() #else GEMAIN() #endif
```

For the first four weeks, I developed the program on the MS-DOS side only. Because of the speed of our tools, a normal compile-link-run GEM cycle would take about two to two-and-a-half minutes. For a program of about 40K object size, this was incredibly useful. It let me add a new routine or function and test each part almost immediately. Productivity in these four weeks was thus incredibly high, and by devoting some very long days, nights and weekends (wow, there goes spring!), I had the basic core finished in four weeks.

Now came the unknown: How long would it take to port it over to the ST? We—mostly Chris—had been working on the ST assembler routines and had already debugged most of ▶

Transportable GEM...

them. Steve had ported most of I*S Talk over by then, and was very familiar with memory allocation and the I/O routines from the Atari developer's document, "Hitchiker's Guide to the BIOS." Because of the way Megamax handles (or rather doesn't handle) assembler interrupt routines, we decided to develop the stand-alone program under Megamax C, and the desk accessory under Alcyon C. Interrupt routines can happen at any time and use any of the 68000's registers. Megamax uses these registers to access GLOBAL variables. Megamax can be used if you keep saving and restoring your registers but, as fast and efficient as Megamax is, the overhead, in this case, was not worthwhile.

THE PORT

With Steve and Chris' help, we began our port one morning at 9:00 am. We were really amazed when, by noon we had the stand-alone program running on the ST. We used Tim Oren's resource mover to smoothly move the resources across, changed our MS-DOS I/O calls to GEMDOS calls, and recompiled. Using Megamax, it all worked first time! (Editor's note: *Tim Oren's resource mover is available for $25. You can reach Tim through his CompuServe PPN# 76703,202.*)

That afternoon, we began porting the desk accessory. This was more of an unknown; the two machines handled desk accessories in different ways. On the ST, a desk accessory is loaded at boot time, while under MS-DOS, it is loaded when you start GEM running. The port was smooth, and by the next evening we had the desk accessory running on the ST. *The main* reason for the delay was the incredibly slow compile and link speed of the Alcyon compiler. The Megamax compiler was about six to seven times faster than the Alcyon.

Over the next four weeks, we were able to fine tune Thunder! to take advantage of all the special features of the ST, including different resolutions, color, and speed. We did this for the MS-DOS environment as well.

At first we used Kermit to port the code, but we have since found an easier method. Just hook up a 3 1/2-inch drive to the PC. (It helps to have someone like Keith Hope for this.) By using any of the versions of MS-DOS made for the new portables (like Toshiba T1100), you format the disk on the MS-DOS side with FORMAT/3, which will make the disk 720K format. Incredibly, we found that the disk is completely recognizable by the ST. Now that better tools exist, you can also port easily with communication software such as I*S TALK, FLASH or HOMEPAK.

With the help of Ian Chadwick's documentation, Marty Herzog's art direction and our in-house desktop publishing system, we had the program completed and packaged by June 1st for the summer CES. Chris finished his function keys and replay routine on both the MS-DOS and ST machines, and they are proving very useful. We have since decided to publish more than twelve ST products this year—by both outside developers as well as in-house.

CRAWLIES AND OTHERS

The most frustrating parts of the exercise were the "idiosyncrasies" (a euphemism for little crawly creatures). Here follow some examples, tips and hints from our experience:

It is hard to keep to the rules when you find that your desk accessory is suddenly 40K bigger than it should be. Yes, it is true. Open a virtual workstation in a desk accessory and it allocates the 300-400 bytes of memory it needs somewhere far away from where it should. A temporary solution is only to open the workstation when necessary.

If you use a regular **form_do** in a desk accessory, you may find that your keys fall through to the application, so be prepared to write your own. By "falling through", I mean that if

I did not want to develop "just another spelling checker," so I began planning the ideal writer's tool for the ST.

you have a desk accessory on top of a word processor, for example, keys pressed during a form in the desk accessory are picked up and used by the word processor.

evnt_timer is a very dangerous call to make from a desk accessory. It sometimes causes the entire program to freeze up. (This one is true under MS-DOS as well, which has its own big list of idiosyncrasies.)

Before doing an **fsel_input** call—and before any I/O—make sure you have correctly set path names. (A: is not good enough, it must be A:\.)

The documentation for **vex_motv** is wrong—it requires a jump to the saved address to update the driver.

form_alert strings can only be about 30 characters per line.

You should modify the source of **form_do** (**form_keybd**) to change underscores to dashes. Underscores blow your application out of the water.

appl_play and **appl_record** do not work without a ROM patch from Atari. (Editor's note: *According to Atari, this code is*

available to developer's only. Contact Atari on the Atari 16-bit SIG of CompuServe.)

COMPILER PROBLEMS

Alcyon: **appl_init** returns the wrong value (always returns 1). Bindings for **form_keybd** and **form_button** are missing

Megamax: No overlay linker support.

The most irritating part of the two compilers are that they have incompatible file formats for object code.

AND YET MORE CRAWLIES. . .

At June CES, we showed the first version of Thunder! before actually shipping it. This was fortunate, because Murphy struck again. We discovered that the program would suddenly freeze up and the menu would stop working altogether. After some frantic tests, we found that the timer problems were even more serious than we thought. Not only should you not do any **evnt_timer** calls, you should not use **MU_TIMER** in your **evnt_multi** calls or have any reference to timer calls anywhere in a desk accessory. We replaced all these references with different **evnt_multi** calls and everything became stable once again. And then, another surprise. The newer versions of 1ST Word do not allow the Backspace key to buffer characters. (When you hold down the key, it does not auto repeat). I presume they did this because the machine is so fast the user might delete too much of the document. This caused problems in the speed that Thunder! deleted characters, so we added an option to delay backspaces for such problems. (Who would believe that someone would write software to slow down the hardware!)

As time passes, it makes less and less sense to develop ST software on an IBM (unless you are developing for both machines simultaneously). As tools on the ST become more mature, and developers become used to the idiosyncrasies of the machine and the tools, (and these idiosyncrasies become documented), it becomes easier to use the ST for its own development. The tools are almost available (the new Megamax compiler compiles faster than my Compaq in 286 mode). Soon, we can expect other tools like Lint, some good debuggers, and, most importantly, the confidence that these tools work as they should. Let us hope our period of frustration is almost over. The time for development of good application software for the ST is here. Only, please, make the effort for us all—and make it good GEM-based software. ■

Batteries Included
30 Mural Street
Richmond Hill, Ontario L4B 1B5, Canada
(416) 881-9941
CIRCLE 203 ON READER SERVICE CARD

NOW ON ATARI ST

MODULA-2
Building with "Software Chips"

De-veloped by Professor Nicklaus Wirth, father of the UCSD Pascal language.

De-signed to encourage the user to write in modules to make programs easy to design, write and maintain.

Full GEM interface and graphics support with access to GEM DOS, GEM AES and GEM VDI.

Also available: Andra (a document processor) and UCSD Pascal software.

Complete with full screen editor linked to the compiler for rapid error detection.

TDI Software, Inc.

Contact your nearest Atari ST Dealer or call: (214) 340-4942

Suggested Retail $79.95

TDI Software, Inc., 10410 Markison Rd., Dallas, TX 75238
Atari is a trademark of Atari Corporation. GEM is a trademark of Digital Research, Inc. UCSD Pascal is a trademark of the Regents of the University of California

CIRCLE 120 ON READER SERVICE CARD

ABBY'S

DISCOUNT SOFTWARE
37 S. Broad St.
Fairborn, Ohio 45324
ORDERS ONLY PHONE: 1-(800)-282-0333
CUSTOMER SERVICE & OHIO RESIDENTS: 1-(513)-879-9699

ATARI ST HARDWARE
520 ST, CPU, W/MODULATOR
MOUSE & SOFTWARE
SF 354 3½" SS/DD DISK DRIVE
SF 314 3½" DS/DD DISK DRIVE
SM 124 12" MONOCHROME MONITOR
SC 1224 12" RGB COLOR MONITOR
SH 204 20 MEG. HARD DRIVE

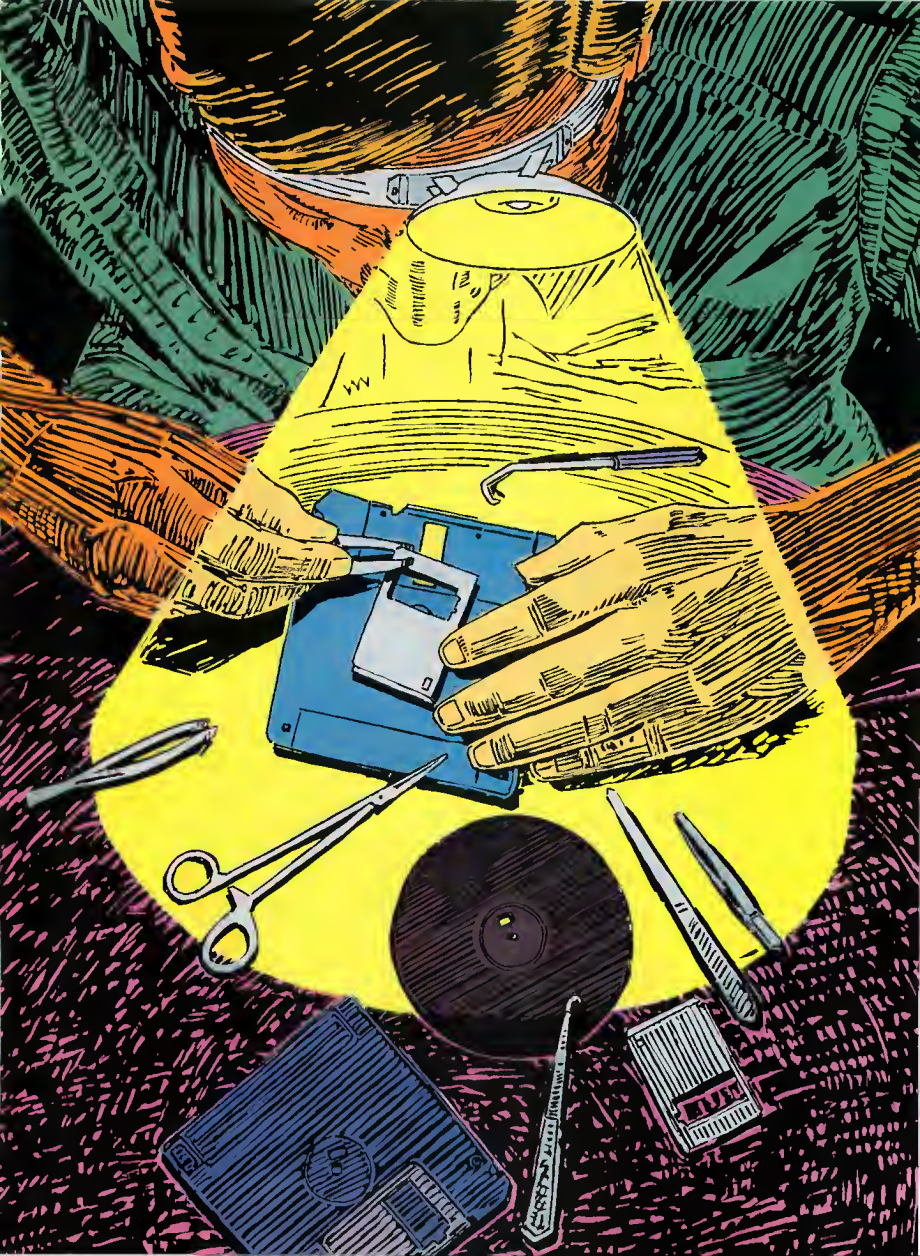
CALL FOR PRICES!!
INDIVIDUAL ITEMS OR PACKAGES

ST TOP HITS

Bally Hoo	$ 29	N Vision	$ 29
Black Cauldron	$ 29	Pawn	$ 34
Borrowed Time	$ 34	Pawnworks	$ 49
Copy II	$ 32	P C Board Designer	$269
Computer Baseball	$ 29	Phantasie	$ 29
Degas	$ 29	Smart Modem	$ 29
Easy Draw	$119	Art Gallery I & II	ea $ 24
Fast/C	$109	Silent Service	$ 29
Fast/Basic M	$ 99	Temple of Apshai Trilogy	$ 27
Fast/Fortran	$109	Thunder	$ 29
Financial Cookbook	$ 39	Typesetter	$ 29
Flight Simulator	$ 29	Universal II	$ 54
Hitchhiker's Guide to the Galaxy	$ 29	VIP Professional	CALL
Homework Helper/Math	$ 35	Winter Games	$ 29
Jet	$ 29	World Games	$ 29
Leaderboard Golf Tournament Desk	$ 29 / $ 14		
Music Box	$ 39	**BULK DISKS**	
Music Studio	$ 35	Gold Star 3½" DS/DD	ea $1 79
		Sony 3½" DS/DD	ea $2 05

SOFTWARE ONLY — Prepaid orders over $50 receive free shipping via UPS in continental U.S. Please add $2 orders under $50. HARDWARE and all orders requiring shipment via U.S. Post Office are subject to additional freight charges. Add $5 for COD orders. VISA/MasterCard orders add $2 service charge. Ohio residents add 5.5% sales tax. Personal checks require a three-week waiting period. No waiting when paid by credit card, certified check or money order. All items subject to availability and price change. WE CHECK FOR STOLEN VISA & MC. ALL RETURNS MUST BE ACCOMPANIED BY AN AUTHORIZATION NUMBER. AO #ATST-886

CIRCLE 002 ON READER SERVICE CARD



PROBING THE FDC

LEARN THE SECRETS OF YOUR FLOPPY

by David Small

David Small, creatar af the MacCartridge, and Atari hacker supreme, waxes eloquent on the subject of ST floppies and the Floppy Disk Controller. This is a thoraugh, technical explaratian af the subject, but even beginners will gain same knawledge. Ta get the full benefit of this article, a familiarity with hardware, hexadecimal, and assembly language is helpful. On yaur START disk, inside the TRAKREAD.STQ falder, is an assembly-language disk reader which demanstrates pragramming techniques for the Floppy Disk Cantraller.

Let's jump right in.

Diskettes are magnetic storage media. In other words, computer bits are stored as magnetic fields on the disk. The actual diskette surface is just iron goo that can be spot magnetized, very much like cassette tape. Unlike cassette tape, which stores music as a varying-intensity magnetic field (analog), the diskette only has two types of magnetic fields which correspond to either a zero bit or a one bit (digital).

The data on the disk is a series of magnetic fields lined up end-to-end. Imagine about 48,000 bar magnets, each with north and south poles, lined up end-to-end around a track and you'll have a fair idea of what the fields look like on a disk. Because some of these magnets are pointed in one direction and some in another, the combination of poles—N/N, S/S, N/S, S/N—creates varying magnetic fields. As the drive head passes over the disk, the head senses these magnetic field *transitions*, or changes, and that is how the data is recorded and reread.

A 3½-inch ST diskette consists of 80 *tracks*. Each track is one circular path around the diskette at a certain radius, and at 135 tracks-per-inch, those tracks are very closely spaced! The outermost track is number zero and the innermost track is number 79.

There is a head in the disk read/write mechanism that moves in and out along the tracks. Once the head is settled on a given track, the disk spins underneath but the head does not▶

WARD SCHUMAKER

move. To read a certain place on the disk, the head moves to the correct track (radius), and waits for the data to spin under the head.

It takes three milliseconds to move the head one track. This is called *stepping* the head, and is done by the *stepper motor*. When it gets to the destination track, it takes 30 milliseconds for the head to quit rattling enough to do something useful. This is known as *head settling time*.

Each track is about 6,000 bytes "long" I say long because, although they are on a circular surface, the tracks do have a beginning and an end marked by an *index pulse*. Every time the disk spins, the drive briefly recognizes one point on the disk which generates a pulse once per rotation. (On 5 1/4-inch disks, a small hole toward the center of the disk is uncovered and a light generates the signal. This same thing is accomplished mechanically on 3 1/2-inch disks by a rectangular hole in the metal hub of the disk.) The disk rotates at 300 RPM (plus or minus one percent), hence there are five index pulses per second, each of which marks the beginning and end of a track.

Typically, the track is divided into *sectors*, which are logical-sized chunks of track. This way you can work with one chunk (sector) of the track instead of modifying the whole thing at once. The Atari ST has nine 512-byte sectors per track, numbered one through nine. Remember, tracks are numbered starting at zero; sectors begin at one.

On most computers, a chip called a *Floppy Disk Controller* (FDC), which sits between the Central Processing Unit and the floppy disk, governs the floppy drive. Think of the FDC as a slow, sophisticated, slow, helpful, slow, and advanced slow chip. If you are programming the FDC, always bear in mind the *slow* part and you'll save yourself a great deal of trouble.

THE CONNECTIONS

To understand the FDC, you must understand exactly what it does to the disk drive. The FDC Track 00 bit, for example, is not going to make much sense unless you've read about the Track 00 detect wire. Since you're using the FDC to talk to hardware, it helps to know the hardware.

The FDC chip used by Atari is the Western Digital 1772. The 1772 is connected to the floppy disk drive with a 14-pin cable. This 14-pin cable expands to a 34-pin ribbon cable inside the drive case. The 34-pin connector has long been an industry standard in 5¼-inch disk drives. Even IBM uses something very close to it, differing only in two wires. Hence, the signals are pretty well standardized. (This is why you can hook a 5¼-inch drive to the Atari ST. See my article in Antic's ST Re-

FDC PINOUTS

FDC TO DRIVE:

6: DRIVE SELECT D. (Not used on ST).

10: DRIVE SELECT A—NOT. (+0 volts=turn drive & light on. Note this doesn't turn on the motor; see pin 16.)

12: DRIVE SELECT B—NOT. (Not used on the ST).

14: DRIVE SELECT C—NOT. (Not used on the ST).

16: MOTORON-NOT. (+0 volts=start spinning the diskette).

18: STEP DIRECTION. When you step the head, this line must tell the drive whether to step in or out. See wire 20.

20: STEP-NOT. This line is BRIEFLY brought to +0 volts to step the drive one track in the direction STEP DIRECTION specifies (in or out).

22: WRITE-DATA-NOT. This is actual bit stream headed for the disk track at around 100,000 baud.

24: READ/WRITE-NOT. When +5, the drive is reading. When +0, the drive is writing.

32: SIDE SELECT-NOT. Pull to +0 volts to write to back side of diskette.

DRIVE TO FDC:

8: INDEX PULSE-NOT. Goes to ground BRIEFLY each rotation, five times per second (300 RPM). Otherwise +5.

26: TRACK 0—NOT. +5 unless drive is at track 0, when this pin goes to +0 volts. This is how the drive tells the FDC to quit stepping it towards track 0.

28: WRITE PROTECT-NOT. +0 volts if the write protect tab is set on the diskette; +5 volts if it is okay to write to the diskette. Note: the drive mechanism will prevent writing if this is set even if the FDC tries.

30: READ-DATA-NOT: The actual real live bitstream from the diskette, at 100,000 baud, complete with wow and flutter (the factory options).

FIGURE 1

source, Nov. '85, titled "ST Uses IBM Disk Files".) Typically the 34 pins are brought out in a ribbon cable, then to an edge connector.

Fortunately, we don't have 34 separate signals to worry about. Every other wire is either Ground or +0 volts, which helps suppress some of the radio noise the cable puts out. There are only 17 active wires, and the wires are in two groups: FDC talking to disk, or disk talking to FDC. No single wire is bi-directional (see *Figure 1*).

The voltages on the wires are always either +5 volts (logic one) or +0 volts (logic zero). Generally, when there is some sort of logical function, the voltages work the opposite of what you would expect: +5 volts means "don't do anything;" +0 volts means "begin." The reasons for this are lost in antiquity.

When you see a wire labelled XXX-NOT, it means that when the wire hits +0 volts (NOT), something happens. This is as opposed to *doing* something at +5 volts, which is more standard *inside* computers. Example: When the MOTORON-NOT line is at +5 volts, nothing happens, but when it is at +0 volts, the motor spins.

SECTOR THEORY

Having discussed diskette theory and the basic electrical connections, let's talk about sector theory.

The FDC must handle the extremely fast data streams to and from the diskette. Once the disk is formatted—all in one long write operation—any disk I/O is done at the sector level in short bursts. If you are to become an FDC Wizard (and you will), it will be your job to properly program the FDC so it can handle those bursts.

Sectors are *read* or *written*, one sector at a time, in discrete chunks. In other words, there is no way to read or write only one part of a sector. The sector is the smallest "addressable" chunk on a disk.

Why 512 bytes per sector (BPS)? Other convenient sizes are 128 BPS (as in the Atari 810 and 1050), 256 BPS (as in many aftermarket 8-bit drives), and 1024 BPS. Atari settled on 512 for the ST because IBM did it that way, and Atari wanted the ST drives to be disk compatible with the new IBM laptops. (And they are.)

When the diskette first comes from the factory, it contains nothing but random magnetic noise. To use it, you must first lay down some marks that are never changed. These mark the beginning and end of each 512-byte sector, and are called *address marks*. This happens each time you format a disk. Think of it as drawing ruler lines on a blank piece of paper.

Address marks consist primarily of *sector headers*, which are followed by *data fields*. There is one sector header and one data field per sector, and nine of them per track, numbered one through nine. Every sector has a six-byte header which identifies the sector ▶

TRACK LAYOUT

Index Pulse
(beginning of the track)

	number of bytes	data (hex)	
	80	4E	Data separator synchronization only
Appears once per track far indexing.	12	00	Data separator sync alas
	03	F6	"FDC, lock up with me to start."
	01	FC	Index Mark
	50	4E	More data separator sync bytes
	12	00	Help the data separator . . .
	03	F5	More sync . . .
	01	FE	"I AM A SECTOR HEADER"
	01	Track #	The track this had better be
	01	Side #	The side number this is (0 or 1)
Repeated nine times per track producing nine sectors.	01	Sector #	The sector number
	01	02	Sector size: 02=512, (00=128, 01=256, 02=512, 03=1024)
	02	CRC	The Sector Header CRC
	22	4E	Resync Data Separator
	12	00	Resync Data Separator
	03	F5	Resync Data Separator
	01	FB	"I AM THE SECTOR'S DATA FIELD"
	512	Data	The actual data bytes (gee whiz!)
	02	CRC	The CRC of the data
	54	4E	Resync Data Separator

Note that the last 54 4E's become the first 4E's of the next sector, immediately preceding the twelve 00's.

FIGURE 2

number (one through nine) that will immediately follow it. The first four bytes of the header are:

Byte	Western Digital Name (ST)	NEC Name (IBM)
1	Track number	Cylinder number
2	Side number	Head number
3	Sector number	Record number
4	Sector Size	Number

As the above table demonstrates, manufacturer terminology is often inconsistent. You'll find "Sector not found" errors and "Record not found" errors. No matter, they're the same thing.

The last two bytes of the sector header are known as a CRC (Cyclic Redundancy Check). There are actually two CRC's per sector. This first one—the header CRC—is a special summation of the previous four bytes of the header. The header CRC is calculated only once, at format time, and written to the disk. Anytime the header is reread, the CRC is checked by the FDC on the fly and compared to what is on the disk header. Any difference means that data on the disk changed without the FDC's help. This translates into a disk error.

After the sector header are 512 bytes of data (the data field) followed by a second CRC that checks the preceding 512 bytes of data field. The data CRC is recalculated and rewritten everytime the FDC writes the sector data.

Although the data field is rewritten many times, the sector header is only written once at formatting time. The FDC handles the timing of these operations.

THE GUTS OF A TRACK

What does a track look like at the byte level? A quick way to find out is dump the entire track to memory, using the FDC Read Track command. Alas, there's a catch. Some of the bytes written to a track at format time can never be reread because they are intercepted by the data separator and never seen by the FDC.

Check out *Figure 2* for the layout of a track. At the end of the track, there is actually room for one more 512-byte sector, which would give you 80 tracks times 512 bytes, or 40K more storage per disk side. The ST can access this "extra sector," but since the name of the game is IBM compatibility, nine sectors per track is standard.

So, if we dump a track using the FDC's Read Track command, we will not see the clean, well-mannered listing in *Figure 2* because those $4E's will be in use by the data separator as it quickly tries to stay in step with the disk. Data won't wait. As the data separator drifts, you will get trash (weird-looking bytes) in the Read Track data. Commonly, the data separator will accidentally lock onto the data bits instead of the clock bits, and the FDC will see lots of $FF's. That's okay, by the end of the $4E field, the data separator should have adjusted itself back into sync. But be forewarned: the data will not appear cleanly and will vary even from read to read.

The bytes in *Figure 2* that begin with "F" do not appear on disk the way they appear in the figure because, when formatting the track, the FDC translates these bytes into missing clock pulses that tell the FDC this byte has a special function—be it a sector header, start of data mark, index mark, CRC, or whatever. You won't see these on a track dump because the FDC will be busy.

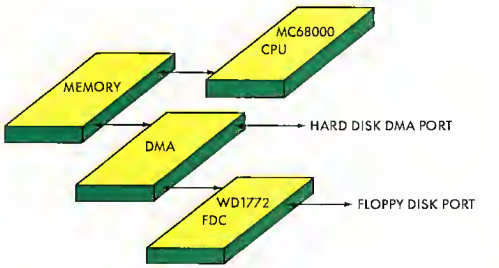


FIGURE 3

OPERATING THE FDC

Programming the FDC involves setting up the TRACK, DATA, and SECTOR registers of the FDC, then writing to the COMMAND register. Just doing the write will cause the command to exe-

cute. You then wait for the FDC to tell you it is finished by watching the IRQ (Interrupt ReQuest) line via the MFP (MultiFunction Peripheral controller) chip, bit five.

Some miscellaneous considerations: You must select the disk-drive number and disk side (done with the Yamaha sound chip, port A, lowest three bits), and you must deal with the DMA controller. But basically, it's a matter of getting the drive online, then writing to the COMMAND register of the FDC to get it to do something.

The TRACK register is used a great deal (Step commands and Sector commands), the SECTOR register is only used for sector I/O, and the DATA register is only used for Step commands because, during sector I/O, the DMA chip handles the data transfer.

THE DREADED MOTORON KLUDGE

Look again at the pinout listing in *Figure 1*, and you will see that one of the wires going from the FDC to the disk drive is MOTORON-NOT. When this line is brought to zero, the motor spins.

There are several subtle considerations here:

1. MOTORON is separate and distinct from DRIVE SELECT. You can select a drive, have its light on, and not have it spinning; or you can have it spinning but not selected. Just because they have always worked together doesn't mean they have to.
2. You must give the motor time to bring the drive up to speed before you do any reading or writing. The data separator is painstakingly tuned to data coming in at 300 RPM. Vary that, and the separation will fall apart. Generally, you need at least 1/5 of a second to get the motor up to speed.
3. You need to shut the motor off when you're done so the drive isn't spinning all the time. Otherwise, the head will literally wear a groove through your diskette. On the other hand, if you have a string of read/write operations, you don't want to shut the motor off after each one or you'll have to wait for the motor to get back up to speed before doing any new operation.

Normally, Western Digital chips require external hardware to accomplish the above feats. To process a command, you switch on the drive motor and wait for it to get up to speed. When done, the hardware keeps the drive spinning another couple of seconds, then reluctantly shuts it off.

Atari set up a flag, called "flock," which may be found at $43E in scenic TOS.

Western Digital, in their infinite wisdom, gave us some hardware inside the 1772 that helps with this task. If you set the Motor-On flag (see *FDC COMMANDS*, below), the FDC will turn on the drive, then wait for several index pulses before continuing; since each pulse takes at least 200 milliseconds, the drive ought to be up to speed. When the command is done, if ten index pulses go by, the FDC shuts off the motor.

The problem is this: You must have a *selected* disk out there before the FDC can see index pulses. Therefore, you must leave the drive selected after completing a command or the FDC won't be able to shut off the motor. But after the motor shuts off, courtesy of the FDC, the drive still stays on. And that's bad news.

Atari solved this dilemma by looking at the drive each vertical blank (VBL), which is 50 times per second in color and 70 times per second in monochrome. If the FDC has just shut the motor off, the VBL task shuts off the DRIVE SELECT, turning off both the drive and its light.

Ah, but to sample the FDC, the VBL task has to *talk* to the FDC, which involves talking to the DMA chip first. All this boils down to big trouble if you're in the process of talking to the disk. So Atari set up a flag, called "flock," which may be found at $43E in scenic TOS. Flock must be set anytime you're going to use the FDC. It tells the VBL task—which will certainly interrupt your program a lot—not to touch the FDC for a while. After you are done with the FDC, turn flock off, and the VBL will shut off your disk drive for you.

If you forget about flock, you're doomed in two ways: The FDC will be stolen from you periodically, or the drive will forget to shut off. Neither alternative is good.

FDC COMMANDS

Let's go through the commands that you can issue to the FDC. For extensive electronic specifications, I recommend picking up the Western Digital 1772 product sheet which can be obtained through most electronic parts suppliers.

The FDC has four *registers*. Registers are a bit like memory. You can write a byte into them, or read a byte out of them. Of course, trying to figure this out from the product data sheet is nearly impossible because they write the sheet▶

FIGURE 4

COMMAND SUMMARY

TYPE	COMMAND	BITS 7	6	5	4	3	2	1	0
I	Restore	0	0	0	0	h	V	r_1	r_0
I	Seek	0	0	0	1	h	V	r_1	r_0
I	Step	0	0	1	u	h	V	r_1	r_0
I	Step-in	0	1	0	u	h	V	r_1	r_0
I	Step-out	0	1	1	u	h	V	r	r_0
II	Read Sector	1	0	0	m	h	E	0	0
II	Write Sector	1	0	1	m	h	E	P	a_0
III	Read Address	1	1	0	0	h	E	0	0
III	Read Track	1	1	1	0	h	E	0	0
III	Write Track	1	1	1	1	h	E	P	0
IV	Force Interrupt	1	1	0	1	I_3	I_2	I_1	I_0

FLAG SUMMARY

TYPE I COMMANDS

h = Motor On Flag (Bit 3)

h = 0, Enable Spin-Up Sequence
h = 1, Disable Spin-Up Sequence

V = Verify Flag (Bit 2)

V = 0, No Verify
V = 1, Verify on Destination Track

r_1, r_0 = Stepping Rate (Bits 1,0)

r_1	r_0	WD1772-02
0	0	6 ms
0	1	12 ms
1	0	2 ms
1	1	3 ms

u = Update Flag (Bit 4)

u = 0, No Update
u = 1, Update Track Register

TYPE II & III COMMANDS

m = Multiple Sector Flag (Bit 4)

m = 0, Single Sector
m = 1, Multiple Sector

h = Motor On Flag (Bit 3)

h = 0, Enable Spin-Up Sequence
h = 1, Disable Spin-Up Sequence

a_0 = Data Address Mark (Bit 0)

a_0 = 0, Write Normal Data Mark
a_0 = 1, Write Deleted Data Mark

E = 15ms Settling Delay (Bit 2)

E = 0, No Delay
E = 1, Add 15ms Delay

P = Write Precompensation (Bit 1)

P = 0, Enable Write Precomp
P = 1, Disable Write Precomp

TYPE IV COMMANDS

I_3-I_0 Interrupt Condition (Bits 3-0)

I_0 = 1, Not Used
I_1 = 1, Not Used
I_2 = 1, Interrupt on Index Pulse
I_3 = 1, Immediate Interrupt
I_3-I_0 = 0, Terminate without interrupt

Courtesy of Western Digital Corporation

primarily for PC designers. So when they say "CS-NOT=0, A1=1, A0=1, DATA register selected," they are *really* saying that if you read address xxx, you'll get what is in the DATA register. The registers look like this:

#	READ	WRITE
0	STATUS	COMMAND
1	TRACK	TRACK
2	SECTOR	SECTOR
3	DATA	DATA

Hence if you read Register 0, you will get the current FDC status. If you write Register 0, you will issue the FDC a command. Registers 1 or 2 are always used to set a track or sector number. Register 3 is used to read or write a byte of data from the controller.

Now, if we were on a normal computer system, there would be four addresses we could directly PEEK or POKE to twiddle the floppy controller chip. We are not. Atari in its wisdom set the DMA chip right smack between us and the FDC (see *Figure 3*). So, not only do we have the exciting thrill of hassling with the FDC, we also have to beg the DMA chip to let us talk to the FDC. We'll solve that in a bit. Take a look at *Figure 4*.

There are four flavors of FDC commands:

Type I commands are essentially "move the head." You will be surprised at the variety of ways Western Digital found to move it.

Type II commands are Read/Write sector commands. There are only two, praise be.

Type III commands deal with the disk at the track level. They are for reading a track, reading the ID marks, or formatting a track.

Type IV commands are Force Interrupts. These are *only* used when the

FDC has gone to sleep and cannot be awakened. However, do not rely on them too much; there are ways to put the FDC to sleep that are permanent until poweroff, and one of the easiest is an improper Force Interrupt. Generally a Force Interrupt is handy during a frozen Read/Write command.

Taking the command types one at a time, I will show the bit pattern for the command, and talk about what the bits do

TYPE I COMMANDS:

Command	Bit Pattern (7–0)
Restore:	0000 h V r_1 r_0
Seek:	0001 h V r_1 r_0
Step:	001u h V r_1 r_0
Step In:	010u h V r_1 r_0
Step Out:	011u h V r_1 r_0

Restore moves the head to track 0. Seek moves to a given track number. Step In and Step Out step the head once in the direction indicated, and Step steps the head one track in the direction previously established by Step In or Step Out. The individual bits are as follows. The function is performed if the bit is ON:

u: Update the Track Register. Reflects the current track. This means you will be able to read the Track register from the CPU to find out what track the FDC is on.

h: Motor-On Flag. This means that before anything happens, the FDC will turn on the drive's motor and wait six revolutions of the diskette to let the diskette get up to 300 RPM before anything happens. You will also find h on other commands. It saves the programmer the agony of wondering if the disk is up to speed.

V: Verify Track. When the step is completed, the FDC will read a sector mark from the disk and compare the track number written on the disk with what it thinks the track number should be. If they don't match, either the disk is weird or the FDC is out of whack with the disk. Caution: a strange disk (such as a copy protected disk) will often have weird track numbers deliberately written into the sector marks. And a disk that is blank will have no track numbers at all.

r_1, r_0: Step Rate. The floppy-disk drive head can only move at a certain speed between tracks; r_1, r_0 selects this speed. It works like this:

You must wait about 32 microseconds after doing a Force Interrupt while the FDC wobbles drunkenly awake.

- **00**: 6 msec per track (use this for 5 1/4-inch disks)
- **01**: 12 msec (no need)
- **10**: 2 msec (kind of fun for 3 1/2-inch, but not reliable)
- **11**: 3 msec (the usual for 3 1/2-inch, unreliable for 5 1/4-inch)

The specifications given in the Developer's Toolkit from Atari are just plain wrong about these step rates. Use the above figures. Generally, stick with 11 for 3 1/2-inch drives and 00 for 5 1/4-inch drives.

Typically, you don't ever use the Step, Step In, or Step Out commands. Use Seek and maybe Restore.

Seek works like this: The FDC thinks it is on track #(TRACK register). It may be wrong because of switched drives or other confusing phenomena. If it is wrong, best correct its thinking by stuffing a number into that register before the Seek. Then, put into the DATA register (#3) the number of the track to which you want to go. Then, issue the Seek command and wait around until the Busy bit of the STATUS register goes away. The FDC is done.

Restore is another handy command. It cares little about where the head is, but pulls the head in until the Track zero line goes low. Restore is helpful after a read/write error because if there's dust on the head it cleans the dust off by moving the head a long way across the disk. Also, if the disk head has wobbled, Restore gets the FDC's TRACK register and the actual head back together.

Something Western Digital didn't think of: There are two drives and one TRACK register. The drives are usually on different tracks. Hence, *anytime* you switch drives, you must store the current track for this drive somewhere, fetch the current track for the new drive from somewhere else, and put it into TRACK. Otherwise, the FDC will get out of sync with the new disk. Atari BIOS routines handle this by reserving some locations for the stored track number. Look at $A08 and $A0C on ROM-based ST systems.

TYPE II COMMANDS

Fortunately, there are only two commands to worry about here. They are Read and Write Sector. They look like this:

Command	Byte# (7-0)
Read Sector:	100m h E 0 0
Write Sector:	101m h E P a_0

Our old friend **h** is still here. Use him if you're not sure if the drive is up to speed, which is *almost always*. The new bits are:

m: Multiple Sectors. Usually you don't do this. If 1, the FDC will read or write sectors starting at the sector number you give it until you Force Interrupt the controller out. This is really a marvelous way to hang the controller. If you are very, very good, you can use this to read an entire track in one rotation of the diskette, like Atari's people did, but beware.

E: Settling Time (30 milliseconds). Assuming you just stepped the head, Setting this bit tells the FDC to wait 30 milliseconds to give the head time to stop rattling--a good idea. (You also can do this during the Step by turning on the verify bit, but that has,other problems.)

P: Enable Write Precompensation. I was hoping not to get into this, but. . .

When you write all those magnetic fields onto a disk, you get the same effect as if you had made a line of bar magnets. North and south poles of two magnets attract, and north and north poles of two magnets repel. This causes your bits to literally wander on the diskette. At read time they can get so far away from where they should be that the data separator cannot follow them.

So, we have Write Precompensation that writes two north-north magnets slightly closer together and two north-south magnets slighly farther apart. This way bits repel, attract, and end up at roughly the correct position. Write Precompensation is not really necessary for single density, but you will want it in a big way for the double density used by the ST. So keep this bit on when writing.

a_0: Data Address Mark. Remember our Beginning of Data Field mark ($FB) which appears at the beginning of each sector's data? (see *Figure 2*). You have the option of using $FB (which you should do), or writing a *deleted data* mark, which just offers a fairly dumb way to mark, at operating system level, whether or not a sector is part of an erased file. If you set this bit, you will write an $F8 instead of an $FB, and every time you read the sector, a status bit will be set telling you this is a "deleted" sector. This causes problems. Summary: Don't.

To read or write a sector, first set the SECTOR register and TRACK register so that they will match the sector and track data found in the sector header. (Otherwise, the FDC will insist "Record Not Found"; it checks both.) Next, issue

FIGURE 5

STATUS REGISTER DESCRIPTION

BIT NAME	MEANING
S7 MOTOR ON	This bit reflects the status of the Motor-On output.
S6 WRITE PROTECT	On Read Record: Not Used. On Read Trock: Not Used. On ony Write: It indicotes o Write Protect. This bit is reset when updoted.
S5 RECORD TYPE/SPIN-UP	When set, this bit indicates thot the Motor Spin-Up sequence hos completed (5 revolutions) on Type I commonds. Type II & III commonds, this bit indicotes record Type. 0 = Doto Mork. 1 = Deleted Doto Mark.
S4 RECORD NOT FOUND (RNF)	When set, it indicotes thot the desired trock, sector, or side wos not found. This bit is reset when updated.
S3 CRC ERROR	If S4 is set, on error is found in one or more ID fields; otherwise it indicotes error doto field. This bit is reset when updoted.
S2 LOST DATA/ BYTE	When set, it indicotes the computer did not respond to DRQ in one byte time. This bit is reset to zero when updated. On Type I commands, this bit reflects the status of the TR00 signol.
S1 DATA REQUEST INDEX	This bit is o copy of the DRQ output. When set, it indicotes the DR is full on o Reod Operotion or the DR is empty on o Write operotion. This bit is reset to zero when updoted. On Type I commands, this bit indicotes the status of the IP signol.
S0 BUSY	When set, commond is under execution. When reset, no commond is under execution.

Courtesy of Western Digital Corporation

the Read command, and watch the fireworks

There are some kludges to go through to set up the DMA chip that I will discuss later.

TYPE III COMMANDS

These are track-level commands. One of them, Write Track—also known as formatting the track—is used all the time. Do this 80 times with a Step in between, and you've formatted the disk!

The Read Address command reads the next address mark on a track and returns it to you. This is a great way for finding out exactly what is on a track, particularly if the track is damaged. If you repeat this command over and over, you'll get a list of all living sectors on the track. Usually this is nine sectors times six bytes, or 54 bytes.

Read Track gives you a fairly accurate dump of the entire track, all 6,000-odd bytes, so have a big memory area waiting to receive data. It generally manages to foul up anytime the data separator resyncs at the beginning of each sector, and sometimes drops a byte out of the blue. It is a handy tool for examining a track that has utterly died, but is not reliable enough for day-to-day use.

Write Track, as mentioned, formats the track. Again, it takes around 6,000 bytes to do this. One way is to set up a memory table 6,000 bytes long and feed it to the FDC as each byte is requested. Atari does it this way. Another method is to set up a bunch of nested loops and do it in code (note how many repeated bytes there are in *Figure 2*). This is fine, but your code has to be darned fast to work, because the FDC will ask for a byte once every 32 microseconds. Even the 68000 doesn't need any distractions when trying to transfer data at that speed, so shut off interrupts while you are doing direct disk I/O. By the way, the DMA controller handles I/O on the ST, and is immune from interrupts.

During formatting, the Floppy Disk Controller takes special action on any $Fx byte and writes special marks to the disk. For instance, when you feed the FDC an $FB, it writes an $FB with a clock byte of $C7 instead of $FF. It also starts the CRC calculator running. ($FB is the beginning-of-data mark, remember.) An $F7 input byte results in the FDC writing two CRC bytes with no $F7 in them at all. Hence, remember this rule: Any byte input to the FDC during Write Track (format) that begins with an $F is going to do something weird.

+5 volts means "don't do anything;" +0 volts means "begin." The reasons for this are lost in antiquity.

This is why you can't do a Read Track, then a Write Track with the same data to clone a track. First, Read Track will not always give you the right sector data, and it will definitely screw up the beginnings of sectors. Second, Write Track does funny things with $Fx bytes, including bytes you find in the middle of a sector; you cannot format data into a sector with $Fx's in it. Instead, format a blank track, then copy the data. If you're fast on your feet, you can issue nine Read Sector commands in one track revolution, then nine Write Sector commands in one revolution. Or, if you're brave, you can use the Multiple Sector Read/Write.

Follow the formatting command with a Read Sector for all the sectors on the track to see if there are any bad spots on the disk. A CRC error means a bad spot.

According to the Atari BIOS manual, formatting the data fields with zeroes is bad karma to the operating system. Besides, it doesn't test the disk very well. Use a repetition of $6DB6 instead. This places maximum stress on the north-north and north-south magnetic field patterns. The operating system will take care of initializing the sectors for boot, directory, and so forth after formatting. Read a freshly-formatted IBM disk sector and you will find $6DB6.

TYPE IV COMMANDS

There is really only one Type IV command: Force Interrupt. Western Digital gives you a bunch of options on how to interrupt the controller (immediately, after index pulse, by next Christmas. . .). Most of these are completely unused. The thing to do is:

Force Interrupt Right Now: 1101 1000, then, 1101 0000. This usually wakes the controller up. (The $D8 is an Immediate Interrupt and the $D0 is a Clear The Immediate Interrupt.)

You must wait about 32 microseconds after doing a Force Interrupt while the FDC wobbles drunkenly awake. Do anything faster and you will shut down the interrupt and really euthanize the FDC.

Do the Force Interrupt whenever anything fails in a floppy disk read/write. Atari does it all the time. It's the "reset1770" subroutine.

STATUS REGISTER

The STATUS register (see *Figure 5*) gives you eight bits of wildly exciting infor- ▶

mation about the last command you did. Generally, it reflects information about stepping commands. But you never can tell. If you need to get the status, do a Seek from the current track to the current track. This is a no-operation command, but it sets the STATUS bits correctly. Atari does this in their BIOS to leave the FDC in a known, Type I status state.

You can read STATUS after a sector read/write to look for deleted data marks, CRC errors, and so forth. But wait 32 microseconds *after* the Busy bit goes away before looking at STATUS. Otherwise, the FDC may not have yet updated the STATUS byte.

The bits look like this:

S7 Motor On
If 1, the MOTORON output to the drive is on. If zero, it is not.

S6 Write Protect
On a Write operation, 1 indicates that Write Protect is active and you can't write to the drive.

S5 Record-type/Spin-up
Type I command: 1 = MOTORON spinup complete.
Type II, III command: 1 = You just read a deleted data mark. Why?

S4 Record Not Found error
1 = You tried to read a sector that was not there. The FDC gave up after 5 revolutions (1 second). If you think the FDC is lying, best check the TRACK and SECTOR registers.

S3 CRC error
A 1 usually means you had a bad data-CRC in the data area; the disk has problems. *However*, if the RNF bit (above) is also set, you have a bad CRC in the sector header, which is usually gruesome to fix. Note that you can often repair a bad data CRC by rewriting the sector. See above discussion.

S2 Lost Doto/Trock 00
Type I command: 1 = You are on track 0.
Type II, III command: 1 = You did not get back to the FDC fast enough to read a byte it wanted to hand you, or you did not give a byte as fast as it wanted. This means your data I/O loop is too slow (I warned you!), or it could mean you forgot to turn the interrupts off and the 68000 got distracted. *Remember*: 32 microseconds per byte, and that means not very darn many 68000 instructions.

S1 Doto Request/Index Pulse
Type I command: This reflects the index-pulse pin. If you read this in a tight loop, it will usually be 1, but every 1/5 of a second it will briefly go to zero.
Type II, III command: 1= The FDC either wants a data byte *right now* or it wants you to read one *right now*. If you're doing "polled I/O," you sit and watch this bit, and whenever it goes high, you read or write a byte.

S0 Busy
1= A command is underway. Don't joggle my elbow. Note that this does not set for 32 microseconds following input of a command, so you had better wait awhile after issuing a command before testing Busy, or it will not reflect Reality.

Generally, you will do a sector read/write, read in STATUS, and mask off the bits that are useless (everything except Lost Data, CRC, Write Protect, or RNF error), then look for any nonzero value equaling disaster.

CORRECTIVE ACTION WHEN READ/WRITE BOMBS

If the read/write bombs, first retry the command up to four times. (CP/M systems retry ten times, which ought to tell you something.) It may work on retry because of dust on the head rubbing off, phase of the moon, electrical transients, witchcraft, or thin spots in the disk oxide.

If that doesn't work, restore the head back to Track zero, then move it back to the track you were working on and try again. This resettles the head on the correct track in case it drifted, and rubs goo off the head. Try this whole procedure twice; it often works. Retries are a fact of life even with diskettes in perfect conditions. Build them into your code and count on them.

THE DMA CONTROLLER

We have mentioned that Atari placed a DMA chip between memory and both the hard disk and floppy disk controller. The DMA chip can kick the 68000 off of memory long enough to do data transfers, and it is essential for the speed required when using a hard disk (see *Figure 3*).

Learn to live with the DMA chip if you want to program the FDC. You can only access the FDC through the DMA chip, and the DMA chip also handles data I/O to the FDC.

To program the DMA chip:

1. Clear the FIFO (First In, First Out), a 32-byte buffer, or "holding pond," in the DMA chip which is used to inhale chunks of data at a time. It must be cleared before new data goes in. (This 32-byte buffer can play hell with the ID-Mark Read command, which only transfers six bytes. They will not make it to memory. After an ID-Mark Read, read in three or more ID Marks to force the data through the buffer.)

2. Tell the DMA chip you're going to be reading or writing by how you do step #1: For Read use $90,$190,$90, and for Write use $190,$90,$190.
3. Tell the DMA chip how many 512-byte chunks you're going to transfer. (This will be sectors in the case of Read/Write Sector, but also applies to track reads of 6,000 bytes.)
4. Ask the DMA chip to let you talk to the FDC registers.
5. Set the Floppy Drive Controller registers (TRACK and SECTOR) as necessary, by first asking the DMA chip to let you talk to a specific register, and then sending data to that register.
6. Ask the DMA chip to let you talk to the FDC command registers.
7. Issue the command to the FDC.
8. Wait for the FDC to finish the command by looking at its IRQ line which goes to zero when the FDC is done. Conveniently, you can sample IRQ with the MFP chip (bit 5).
9. You're done. So poll the DMA chip to see what it thought of the transfer. Was there a DMA error? No? Good. Bit 0 of DMA STATUS should be nonzero.
10. Ask the DMA chip if you can poll the FDC. Poll the FDC status register to see what the FDC thought. Was there an FDC error? No? Good. The various status bits should all be zero.

You ask to talk to the FDC by putting a special number into the FIFO register of the DMA chip:

$80: COMMAND/STATUS register
$82: TRACK register
$84: SECTOR register
$86: DATA register

Then, put the value you want sent to the FDC into the DISKCTL register of the DMA chip, and it'll make its way to the FDC.

The FDC is just not fast enough to keep up with the 68000 on anything, so you must surround all accesses to the FDC with delay loops. You will find these at the bottom of listing for the program, TRAKREAD, enclosed on your START disk. They can be most valuable.

ATARI'S FDC BIOS

Atari wrote a great deal of software to run the floppy disk. It is sent to all developers as FLOPS in the BIOS listing. Because you must access the FDC through the DMA chip—which is an Atari-only part—an examination of this software is essential. There is little or no documentation about the DMA chip. The only place you can look to learn is the floppy disk driver in the BIOS.

Atari in its wisdom set the DMA chip right smack between us and the FDC.

The TRAKREAD program on your START disk uses Atari's own BIOS routines. (Atari gave me permission to reprint them, thank heaven.) I weeded out things that were not essential for the task at hand (such as the Vertical Blank handler) and shortened, straightened, and commented the code for your reference.

I use Atari's calling protocol for everything because, when you switch from this program to the real BIOS, everything will remain consistent.

I've also modified Atari's Read-Sector routine to do Read-ID Mark and Read-Track, then called the Read-ID and Read-Track. The calling sequences tell all about the request, and can be modified if you should wish to.

THE PROGRAM

TRAKREAD is written with AS68, Atari's very own assembler. It should be compatible with most in-line assemblers available with the various C packages. Using it is easy. Double click on TRAKREAD.TOS, then insert the disk you wish examined. At the prompts, choose drive A or B; side Front or Back (use Front on single-sided drives); enter the Track number in hex (from $00 to $4F), and the disk will be read. The next prompt will ask you where to dump the information (C for console, P for printer). During the dump, you may press any key to pause; press Control-C to exit to the Desktop.

TRAKREAD does some obvious things, such as using Read-ID and Read-Track commands to scan the ID address marks and data on the track. But it also does a host of less visible things, such as stepping the disk. Take the program apart and examine it closely. It is provided primarily as a concrete example for all the theory we've been discussing. We've covered a great deal of material in this article. I don't know about you, but I'm happy to bring it to a close. ■

NEW for the ST . . . from MichTron

CORNERMAN $49.95

What Sidekick did for the IBM, *Cornerman* does better for your Atari ST! This utility gives you a host of useful desk-top tools in one simple, neat package. With everything from a built in clock to a full function calculator, a cluttered desk is a thing of the past. And as a Desktop Accessory, *Cornerman* is a available nearly anywhere within GEM.

- **REAL-TIME CLOCK** in digital or analog form.
- **NOTEPAD** lets you write and store notes for handy future reference and reminders.
- **ASCII TABLE** shows the ST's symbols with their decimal & hex values. Great for programmers!
- **PHONE BOOK** stores all your important names & numbers, dials through your modem and even records elapsed calling time!
- **16 DIGIT CALCULATOR:** works in binary, octal, decimal, and hex; 3 memory registers; math and logic functions; base conversions; "tape" printer output.
- **DOS WINDOW** for easy access to MichTron's Dos Shell program.
- **15-SQUARES GAME** simply for entertainment.
- Doesn't interfere with other programs.

MICHTRON UTILITIES $59.95

What would a utility be worth that could turn back time and end frustration? Before you find out the hard way that such a utility would be priceless, prepare yourself with *MichTron Utilities*. This utility searches for and retrieves deleted files and lost data. Change file contents, attributes, file and volume names, or any individual bytes on disk! Just type the new data or click on selector buttons. You can also format individual disk tracks and copy individual sectors to repair damaged disks. A new 10 sector format utility lets you add 80K of storage capacity to you disks.

Personal Money Manager $49.95

This personal accountant will keep accurate totals for as many expenses, income sources and accounts as you need. Print checks and other kinds of reports for a true picture of your current financial status. You can form projected budgets for future expenses and income. Fast and powerful, the Money Manager uses the GEM system for simplicity.

UTILITIES:

THE ANIMATOR ($39.95) - Animate Neo or Degas pictures for business presentations, or just for fun.
BBS ($49.95) - Complete Bulletin Board System
BUSINESS TOOLS ($49.95) - Over 200 attorney-prepared business forms, letters, and contracts.
CALENDAR ($29.95) - Desktop appointment calendar.
D.F.T. ($49.95) - Transfer files between the ST and IBM.
DOS SHELL ($39.95) - Mimics the MS-DOS command structure: "global" commands, batch files, and more.
ECHO ($39.95) - Uses X-10 modules for a wireless remote-control system for home or office
Introduction To LOGO ($49.95) - Easy tutorial lets you learn to program in Logo.
KISSED ($39.95) - Debugger features full-screen editing, miniassembler/disassembler, help function and more.
M-DISK ($39.95) - RAM-disk emulator gives you the equivalent power of an extra disk drive!
MI-DUPE II ($39.95) - Fast, easy file duplication.
MI-TERM ($49.95) - Advanced communication program lets your ST talk to virtually any other system.
MIGHTY MAIL ($49.95) - Complete mailing-list manager features multi-level sorts and conditional "flags" for specialized reports.
SOFT-SPOOL ($49.95) - Frees your ST to print & compute at the same time.

ARCADE GAMES:

GOLD RUNNER ($39.95) - Infiltrate enemy gold mines. 63 screens test your skill and logic (color monitor).
MAJOR MOTION ($39.95) - Race down the highway, destroying enemy spies as you gain new weapons and defenses (color monitor).
MISSION MOUSE ($39.95) - Avoid the prowling cats as you climb to the next goal (monochrome monitor).
TIME BANDIT ($39.95) - Collect the Treasures of Time! Choose from a multitude of worlds: 13 unique arcade lands and 3 complete arcade/adventures (color monitor).

STRATEGY GAMES:

FLIP SIDE ($39.95) - Play Reversi against live or computer foes.
CARDS ($39.95) - Blackjack, Cribbage, Klondike, Poker Squares, and Solitaire (medium or high res only).

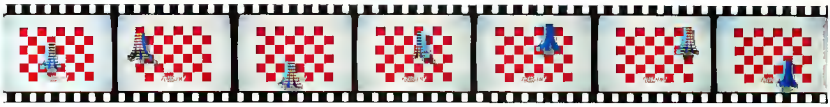
All reasonably priced, with more coming every day. Ask for our latest catalog!

Dealer inquiries welcome • Visa and Mastercard accepted • Add $3.00 shipping and handling to each order.

MichTron

576 S. Telegraph, Pontiac, MI 48053
Orders and Information (313) 334-5700

CIRCLE 084 ON READERS SERVICE CARD



FUJIBOINK!
BEHIND THE BIT PLANES

FIND OUT HOW

by Xanth Park

FujiBoink!, the bouncing roinbow fuji, is the best grophics demo START hos seen on the Atori ST. We hove received mony requests from people wonting to know just how it wos done. So, we went to the source: the progromming wizords ot XANTH. After o little orm twisting, they ogreed to relinquish their secrets. All reloted programs, plus source code, moy be found in the FUJI.STQ folder on your START disk. Pleose note: FujiBoink! requires o color monitor.

In the beginning, there were bouncing balls. Lots of bouncing balls: Amiga, Atari 8-bit, and Atari ST. Then there was FujiBoink!, which demonstrated the superiority of Atari 8-bit graphics. Now, FujiBoink! has been ported to the ST. This is the tale of that heroic feat.

(**Editor's note**: *Amiga introduced its graphics power with a demonstration featuring a rotating bouncing ball. In November, 1986, Atari showed up at COMDEX, in Las Vegas, demonstrating the same bouncing ball on both the ST and the Atari 8-bit machines. The 8-bit version was done by XANTH, a Seattle computer center. A few weeks later, at CES, XANTH was demonstrating a rotating, bouncing fuji (the Atari trademark symbol) with scrolling rainbow on the 8-bit Atari's. Inevitably, this same demo was adapted, with startling results, on the Atari ST.*)

A while ago, a fellow and I were discussing the Amiga and its bouncing ball. He said he was impressed by the large number of rotation calculations that the Amiga was doing in real-time. I hated to disillusion him, but I had to set him straight: The spinning is just an illusion. Now, I don't know what people think FujiBoink! is doing, but again, it's all a trick. If you're expecting some sort of miraculously fast 3D rotation and polygon-filling routines, do not read further—and give my regards to the Easter Bunny.

But before we ruin the illusion by explaining how the magic works, those of you who have not seen FujiBoink! should read the sidebar and get the program up and running. ▶

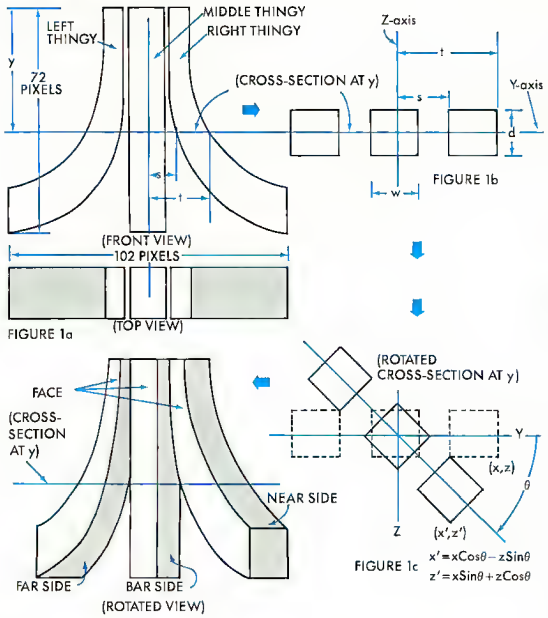


FIGURE 1

XANTH TELLS ALL

The Effect: A red-and-white checkerboard pattern is displayed. A translucent title square fades in over the checkerboard. The Atari fuji symbol appears, a rainbow of colors cascading down its face. It casts a transparent shadow on the background. The title square fades out. The fuji falls and bounces and spins all over the screen at 60 frames per second.

How It's Done: FujiBoink! sets the screen to low-resolution and calls VDI to draw some red squares on a white background. It reads part of FUJIBOIN.D8A (d-eight-a: data, get it?) and stuffs it into screen memory for the title screen, which is actually a portion of a NEO picture. The light pink squares of the title square create an illusion of translucency, which is heightened by the fade-in and fade-out, accomplished by adjusting the color registers.

The rest of FUJIBOIN.D8A is read in. Most of this data consists of 32 predrawn views of the fuji in varying stages of rotation, and the rest is shading data for the side of the fuji. Successive views are moved into screen memory at the appropriate place and time, thus creating the illusion of rotation and movement. The transparent shadow is the result of careful manipulation of bitplanes and color registers. The rainbow is generated by a raster-interrupt routine.

GENERATING THE FUJI VIEWS

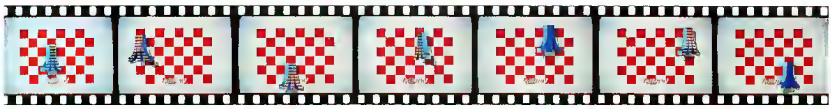
Well, the secret is out: FujiBoink! is just a glorified cartoon. Does this mean you'll stop reading?

Thought not. I will now attempt to explain how the 32 views of the fuji are generated by the FUJIDRAW program.

The fuji symbol has three basic parts, which I call the *left thingy*, the *right thingy*, and the *middle thingy* (see *Figure 1a*). At any value of Y, the fuji's cross-section can be described by the middle thingy's width, W; the fuji's thickness, D; and S and T, which define the contours of the left and right thingies (see *Figure 1b*). The values for S and T were chosen manually, although a mathematical description is certainly conceivable.

To create the data file, a rotation operation is performed on each point of the cross-section and the face, farside, barside and nearside are plotted on the screen (see *Figure 1c*). Then, for the shadow, each thingy-section is projected mathematically against a backdrop, assuming a light source 11.25 degrees to the side and 10 degrees above the horizon.

This basic process is repeated for ev-



ery value of Y and every rotation. Each view, as it is completed, is written out to disk. Actually, only bit-planes 1 to 3 are saved, for reasons soon to be revealed.

One other point: Each view is offset horizontally from the preceding view so that no run-time shifting is required to move the fuji horizontally. As you can see, FujiBoink! proper hardly does any work.

TRANSPARENT SHADOWS AND DISPLAY STRUCTURE

The ST uses a bit-plane display structure. In low resolution, four 320-by-200 bit-planes are used (see *Figure 2*). The principle behind the transparent shadow is this: draw the background grid on plane 0, do all animation in planes 1 to 3 (leaving plane 0 alone), and with the proper color register assignment, voila! Transparent shadow.

Perhaps an example will help to elucidate matters (see *Figure 3*). In this particular case, the shadow of the fuji is represented by pixel 001x binary. What's with the "x"? Well, the shadow is drawn as 0010 pixels, but only the three most significant bits are saved; the least significant bit effectively merges with plane 0, the grid. This way, the color of the shadow will depend upon what it overwrites. The fuji's face is pixel 010x, and empty space around the fuji is 000x.

Now, if at a particular point the grid is 0 (white), the situation will be empty space (0000), shadow (0010), and face (0100). Therefore, color register 0 should be white, register 2 should be grey, and register 4 should be whatever color the fuji's face is supposed to be. If, however, the grid is 1 (red), the empty space pixel will be 0001, shadow will be 0011, and face will be 0101. Color register 1 should be red, register 3 should be a dull red, and register 5 should be the same as register 4, because the color of the fuji's face does not depend on the grid. On the other hand, the colors of empty space and the shadow do depend on the grid, and by using bit-plane 0 to select appropriate color registers, the transparency effect is easily produced.

OVER THE RAINBOW

The rainbow effect, which is so easy on the 8-bit Atari's, posed two major problems on the ST: how to change the fuji's color on the fly, and what to change it to.

The first problem was solved in Tom Hudson's ST Color Palette article (ANALOG magazine, January, 1986). The technique consists of programming the 68901 Multi-Function Peripheral chip to generate an interrupt every scan line. The interrupt service routine can then stuff new color values into the color registers. This is not the way Atari recommends doing this. Their method uses the built-in HBLANK (Horizontal Blank) interrupt. Unfortunately, I was unable to get that method to work, so I left well enough alone, and went to bed.

The second problem was an esthetic one. When the ST's 512 colors are laid out in RGB order (blue varying most rapidly), the rainbow does not really look so hot. The order I finally settled on is generated thusly (in pseudo-pseudo-code): ▶

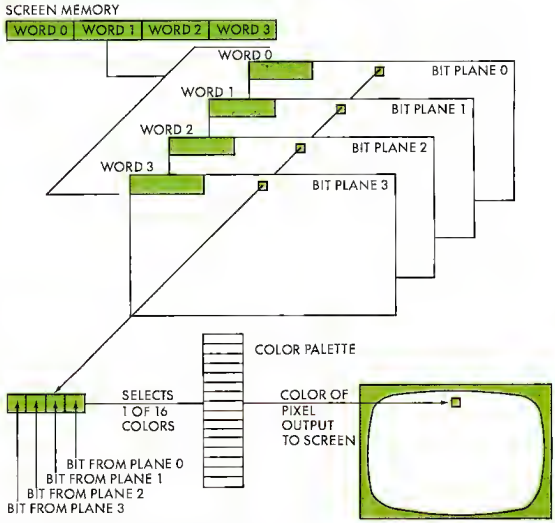


FIGURE 2 *ST low-resolution display*

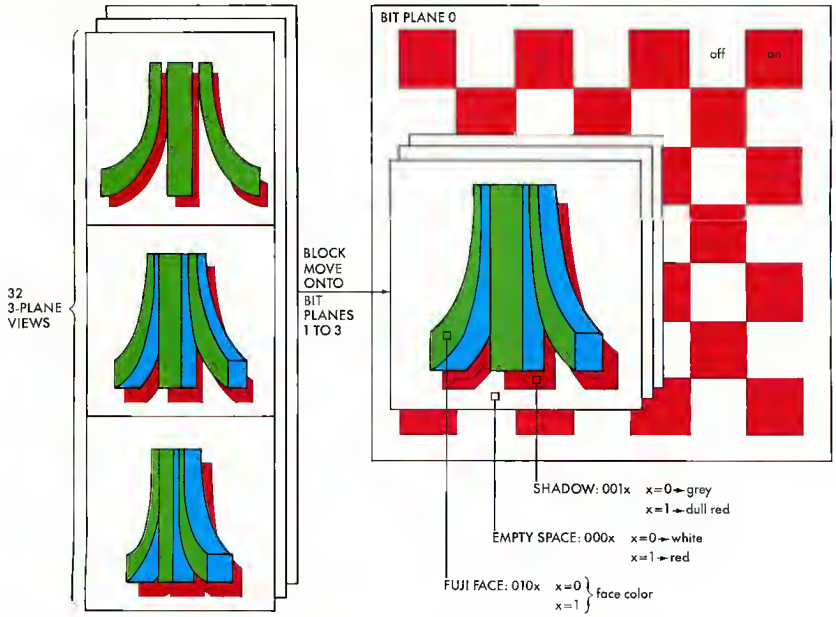


FIGURE 3

```
i=0
 for red=0 to 7
  for blue=0 to 7
   for green=0 to 7
    rainbow( i )=((red*3) mod 8)*256
                 +green*16
                 +blue
    i=i+1
   next
  next
 next
```

The red component is multiplied by three to provide a little more variety in the rainbow.

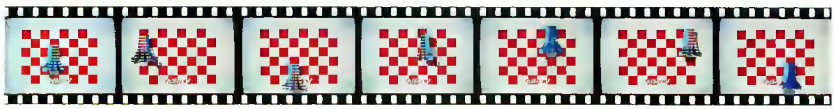
CALCULATING SHADING

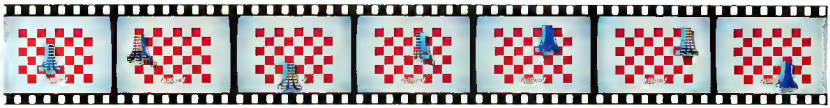
This is embarrassing. This part of the project took nearly four days, mostly because of my own stupidity. (Sample mental process to convert 90 degrees to radians: Well, 90 degrees is one-quarter of a circle, so in radians that's one-quarter pi, right? AAAARRRGH!) This was also the time I discovered that Alcyon C wouldn't let me initialize a **float** array, and that the arc-tangent function **atan( )** wouldn't link. More AAAARRRGH!

The idea was to generate realistic coloring on the side of the fuji, for each scan line of each view, according to the following formula:

brightness=d*cos(i)+s*cos(a)^n

where:





d is strength of diffuse reflection (70%)
i is angle between light source and surface normal
s is strength of specular reflection (30%)
a is angle between reflected light and sight vector
n is 5

I didn't make all that up. I got it out of a book (see *Reference*).

Each brightness value is mapped onto an ST color, starting from dark blue at the low end, through light blue to white at the high end. The interrupt routine described above also stuffs the fuji side-color registers.

THOSE GREAT SOUND EFFECTS

I know, I know, there's only one sound effect, and yes, I agree, it's not so great. Maybe next time.

ETCETERA

FujiBoink! is written mostly in C. The 3-plane block move and clear routines and the interrupt service routines are written in assembly language. Those who wish a detailed, technical explanation of the coding are invited to dissect the source code on the START disk.

(**Editor's note:** *START is interested in seeing any variations on FujiBoink! that readers come up with. How about a custom data generator?*)

REFERENCE:

- *Procedural Elements for Computer Graphics* by David F. Rogers, Osborne/McGraw-Hill, Berkeley, CA. ■

CREATING AND RUNNING FUJIBOINK!

FujiBoink! requires a data file over 118K in size, which is far too large for the START disk. The FUJI.STQ folder holds three programs that create separate portions of the data and a fourth program that binds them into the final data file. The following instructions will get your fuji up and bonking.

1 Open the FUJI.STQ folder on your START disk and copy the following files to a blank, formatted disk.

FUJIDRAW.PRG
FUJISHAD.PRG
GETTITLE.PRG
PEND.PRG
TITLE.NEO
FUJIBOIN.PRG

2 IMPORTANT! Set your ST's resolution to Low. Some of the data creation programs do actual screen grabs.

3 From the disk you just moved the six files to, double-click and run FUJIDRAW.PRG. This program will create a series of fuji's on the screen and write them to a data file called FUJIDRAW.D8A. The whole process will take about seven minutes.

4 Now, run FUJISHAD.PRG, which will create a data file called FUJISHAD.D8A.

5 Next, run GETTITLE.PRG. This will display the TITLE.NEO picture and capture a portion of it to a file called TITLE.D8A.

6 Click and run PEND.PRG. It will bind the other three data files into one 118K file called FUJIBOIN.D8A. Once you have your final FUJIBOIN.D8A FILE, you may delete the other three .D8A files. (PEND.PRG will display a couple of bombs before returning to the Desktop. Try to pretend this didn't happen. It won't affect the final result.)

7 Copy FUJIBOIN.PRG to the same disk as FUJIBOIN.D8A. These are the only two files you need to run FUJIBOINK.

8 You're done! Click and run FUJIBOIN.PRG to see the demo. While running, you may press any function key to freeze or restart the bouncing fuji. Press the Space bar to return to the Desktop.

A-FILER/REPORT WRITER — $59.95

If you can click your mouse you can run this Filer/Report Package

This package is an extremely powerful GEM based mouse driven data base and report writer. The business applications are endless. It has 24 fields, disk based for unlimited number of records, and the ability to do dollar and decimal point calculations. Standard ASCII D-Base and other data base files can be used. The Report Writer can be used for any custom forms like invoices, labels, C.O.D. tags, visa slips, statements and inventory reports, to name a few. It's easy... just grab the field with your mouse, put it where you want it and...click! That is where it will appear on the forms you print.

A-MAILER ST — $49.95

At last a Mailing Labeling program that is not only super powerful, but also Mouse driven and easy to use. This program is NOT Ram based so you can have as many records as your disk drive can hold. One of the many powerful features includes the ability to insert into a field text that was a prior record just by hitting the TAB key. Other features include 9 fields, labels for 1, 2, 3 across, ability to custom size your labels and port over D-Base or other standard ASCII mail lists. This program also integrates with the Graphic Label Maker.

GRAPHIC LABEL MAKER ST — $39.95

This package will allow you to use graphics when printing your address labels. It even gives you different typestyles. This program will interface with both the A-Mailer and XXXXX Rated Graphic Data Disk for those unusual labels.

Utilities available for other systems include Keymaster and ToolBox 64/128 for the Commodore 64/128, A-Copy and Amiga Tools for the Amiga.

COMING SOON: KeyMaster for the Apple, Atari ST TOOLS which will include hard drive utilities.

To place your order by phone **(206) 687-7176** or in **Canada call (705) 746-8081.**

For Mail Orders.

$39.95 plus $3.00 Shipping & Handling

MegaSoft LTD
P.O. Box 1080
Battle Ground, WA 98604

or in Canada: **MegaSoft Canada** LTD
P.O. Box 10
Parry Sound, Ontario 92A 1P8

RAM DISK AND SPOOLER — $39.95

Both the Ram Disk and Spooler are included in one package for one low price. They are both completely transparent to your operation. Ram Disk will even allow you to press the reset button to interrupt a running program and preserve the Ram Disk contents. The Spooler will allow you to continue to use your computer while the printer is in operation.

ST TOOLS — $39.95

ST Tools is a complete utility package for the ST. This package is completely GEM based and mouse driven. ST Tools will allow you to do just about anything with the ATARI DISK DRIVE. Some of the options included in this program are: COPY, MOVE, EDIT, RENAME, COMPARE, UNERASE, ERASE, CHANGE STATUS, CHANGE LABEL, FORMAT, and RECOVER ERRORS. Also included are hard drive utilities.

XXXXX RATED GRAPHIC DATA DISK ST XXXXX — $29.95

These graphics work with your Print Master or Graphic Label Maker ST. There are over 60 entertaining graphics on this disk. They can be included with anything you can make with Print Master and Graphic Label Maker ST. (labels, letterhead, banners, cards, announcements, and many more)

TELECOMMUNICATIONS PACKAGE — $39.95

This package is a total package that includes all the features of the more expensive packages. Options include: spooling, printer dumps, phone book, 300, 1200, 2400 and 9600 baud rate, up/down load, and x-modem with remote access. Included in this package is the Wargames Auto Dialer. Wargames will help find other local computers with modems on them.

Dealer and Distributor Inquires Invited.

Enclose Cashiers Check, Money Order or Personal Check. Allow 2 to 6 weeks for delivery.

VISA

MasterCard

$3.00 S & H on all orders
Software Submissions Invited

To place your order by phone **Call (206) 687-7176**
or in **Canada call (705) 746-8081**

For mail Orders:

MegaSoft LTD
P.O. Box 1080
Battle Ground, WA 98604

or in Canada: **MegaSoft Canada** LTD
P.O. Box 10
Parry Sound, Ontario 92A 1P8

THE WAIT IS OVER!

MT C-SHELL™ IS HERE.

MAIN FRAME PERFORMANCE FOR YOUR ST . . .

- Multiuser and Multitasking
- Electronic Mail
- Print Spooling
- Unix™ Like Environment
- And it Runs TOS Programs

$129.95
COMPLETE

Beckemeyer Development Tools

592 JEAN STREET, #304, OAKLAND, CA 94610
415/658-5318

CIRCLE 008 ON READER SERVICE CARD

ProCopy

INTRODUCTORY OFFER

$24.95 ST BACKUP UTILITY

Regular $34.95 Offer ends July 31, 1986

Software can easily be damaged by heat, humidity, magnetic fields, wear & teor or just dirt. The exposed surface of a disk can be affected by even a touch. The best way ta preserve yaur saftware is to make backups. ProCapy provides the solution.

- Duplicates virtuolly ony protected disk.
- Reosonoble updote policy ovoilable.
- Protects against accidental loss of expensive software.
- Not copy-protected for your convenience.
- ProCopy works on oll Atori ST computers.

ORDER TODAY AT THIS REDUCED RATE

Send $24.95 (check or money order). VISA & MC accepted. Add $2.00 shipping & handling in U.S. & Canada. Overseas shipping & handling add $4.50.

Atari ST is a registered trademark of Atari Corp.

Available Only From

PROCO PRODUCTS
P.O. BOX 665
CHEPACHET, RHODE ISLAND 02814 USA
(401) 568-8459

CIRCLE 091 ON READERS SERVICE CARD

S&S Wholesalers, Inc.

226 Lincoln Road
Miami Beach, Florida
33139

DISKETTES

Winners DSDD 3½"	$25.50
Maxell SSDD 3½"	$18.50
Sony SSDD 3½"	$17.50
Maxell DSDD 3½"	$29.50

SPECIAL

Video Digitizer for the 520 ST/1040 ST $177.77
Convert Video Signecs for Computer Reproduction

ST PERIPHERALS

SF 314 Disk Drive	$192.77
SF 354 Disk Drive	$129.77
SM 804 Graphic Printer	$177.77
SH D 204 20 mb Hard Disk	$627.77

1040 ST

Color & Mono Systems
CALL FOR BEST PRICING

ST MONO SYSTEM

- 520 ST Computer
- 124 Mono Monitor
- SF 354 Disk Drive
- Basic
- Logo
- First Word

$499.77

ST COLOR SYSTEM

- 520 ST Computer
- SC 1224 Color Monitor
- SF 354 Monitor
- Basic
- Logo
- First Word

$737.77

ST SOFTWARE

Kings Quest II	$31.77
Homework Helper Meth	$31.77
Financiel Cookbook	$33.77
Little Computer People	$33.77
Music Studio	$36.77
Mind Shadow	$28.77
Winnie The Pooh	$14.77
Breteccus Checking Account	$27.77
Maneger	$44.77
Rogue	$23.77
D.E.G.A.S.	$23.77
Habba Writer	$43.77
Ultime II	$17.77
Hacker	$25.77
Art Gallery	$18.77
Print Master	$23.77
Borrowed Time	$28.77
Habba View	$43.77

INFOCOM

Zork I	$24.77
Zork II	$24.77
Zork III	$24.77
Suspended	$24.77
Ballyhoo	$24.77
Deadline	$29.77
Starcross	$29.77
Suspect	$27.77
Cutthroat	$24.77
Seestelker	$24.77
Infidel	$27.77
Sorcerer	$27.77
Enchanter	$24.77
Witness	$24.77
Planetfall	$24.77
Spellbreakar	$29.77

EASY DRAW	$94.90
THE PAWN	$27.77

TO ORDER CALL TOLL FREE
1-800-233-6345
BETWEEN 9 A.M.-6 P.M. EST
MONDAY THROUGH FRIDAY
In Florida call (305)536-1364

CIRCLE 110 ON READERS SERVICE CARD

Arick Anders has been developing IBM PC educational products for the past two years with a company called WICAT in Orem, Utah. The touch-typing program he developed is now being marketed by the IBM educational division. Arick received his B.S. degree in Computer Science with a minor in Chemistry from Brigham Young University. On the weekends he is a medic with the National Guard and during his spare time he raises English Angora rabbits, three children and tries to keep up with his wife. She is a registered nurse and works in the emergency room of the local hospital.

Michael Bendio works for Hart Scientific, writing data-acquisition software. He has been a professional programmer since 1978 and received degrees in Comparative Literature from Brigham Young University and the University of Chicago. He's been fascinated by computers since he read the Popular Electronics construction article for the MITS Altair. His first computer was the Processor Technology SOL with 2K of RAM and a binary switchbox for a keyboard.

Christophe Bonnet of France is a 21-year-old university student studying physics, electronics and (surprise) computers at the University of Marseilles. His six years of computer experience have involved work on the Apple II and the Macintosh. He wrote Reversi after some four months with the Atari ST. He is currently at work on a 3D graphics program for the ST.

Christophe also has an interest in war-simulation games and tentatively plans to develop his own, following completion of the graphics project. When the computer is down, Christophe heads for the tennis courts or the slopes. He lives with his family in Marseilles.

Christopher Chabris appeared in the premiere issue of START with "ST Assemblers—A START Comparison." A student of computer science at Harvard University, Christopher maintains a strong interest in artificial intelligence and is completing a book on the subject which will be published in 1987 by Dow Jones-Irwin/Multiscience Press.

Christopher is among the top 50 chess players in the United States under 21 years of age. His knowledge of programming languages includes 68000 assembly, LISP, and PROLOG. Christopher has owned one of every Atari computer as far back as 1980, "with the exception of the 1200XL."

Ian Chadwick has been writing about computers since 1979. His articles have appeared in the Toronto Star, Softside, Antic, Analog, Computer Gaming World, and other publications. He was an editor on the Canadian computer magazine, Info-Age, and was the trade book editor for Holt, Rinehart & Winston of Canada. He is best known as the author of Mapping the Atari, published by Compute! Books, and of software manuals for Antic, Batteries Included and other publishers.

Ian lives in Toronto, Canada, with his wife Susan, their six cats and one dog. He is now writing mystery fiction and trying to teach his dog to play chess.

Jim Dunion was a programmer with the "old" Atari, where he first worked in the Software Development Support Group, then later in Alan Kay's research group. He has been with computers since the first days of microcomputers and was founder of Peachtree Software in Atlanta, Georgia. Presently, he is with The System Works in Redmond, Washington.

Jim designed Dunion's Debugging Tool, considered by many to be the best debugger for the 8-bit Atari, and is now working on an ST version to be called STDDT.

Tom Hudson is the author of DEGAS, CAD-3D, and the new DEGAS Elite. He was head of programming with Analog magazine from 1982 to 1985 before leaving to become a free-lance software developer. In our premiere issue, Tom wrote "Stealing the ST Printer Driver," a multi-printer screen-dump desk accessory.

Tom's first computer was an IBM 1620 that he played with in high school. He worked his way through college teaching computer neophytes, then landed a job as programmer/operator at a savings and loan while earning a BS in data processing. Tom lives in Mission, Kansas and is a SYSOP in the 16-bit Atari section of CompuServe.

Harry Koons first programmed the IBM 704, 709 and 7090 mainframes while attending MIT. There, he received a BA in Physics and a Phd in Geophysics before moving directly to The Aerospace Corporation where he remains employed as senior scientist building instruments

for satellites. "There are about half-a-dozen major satellites orbiting the earth with my equipment on board."

Harry developed several award winning products for the original APX (Atari Programmer's Exchange) line, including Mapware and Starware. His ST programs include Maps and Legends, and The Astrologer (available from The Catalog). Harry is also the author of The Tax Advantage, for the 8-bit machines, and is currently working on a tax program for the ST planned for next tax season.

Alex Leavins built his first computer when he was in the fourth grade. It added, subtracted, multiplied, and divided binary numbers. "It was a real primitive calculator and it was about the size of an Atari 800. But it worked." Alex went on to graduate with a BS in mathematics at the University of Chicago. From there, he worked at Midway Manufacturing, designing Atari 2600 games. He then branched out on his own forming Dynamic Software Design.

Alex's interest in text adventures goes back some years—he coauthored Wombats and Wombats II, two satirical Infocom-style text adventures written for the Atari 8-bit computers. Alex recently joined KnowledgeSet as a software engineer.

Daniel Moore, along with partner Steve Ahlstrom has written several programs for the 8-bit Atari's, including Paperclip and Synfile. Dan recalls his first computer was a Hewlett Packard 2114B in high school: "It was the size of an extra-large. legal file cabinet and contained 8K words of memory. It used paper tape and a 30-line Fortran program took three hours to compile."

Dan attended the Colorado School of Mines. "We were six blocks from a Coors brewery and I remember waking up in the dorm, opening the window. and taking a deep breath of Rocky Mountain hops." Dan is currently working on the Hometext portion of Russ Wetmore's HomePak for the ST, and on Paperclip Elite for the ST.

Tim Oren wrote "Tracking the Elusive GDOS" in the first issue of START. He was one of the original designers of the GEM system at Digital Research, working on the AES portion and sections of the Desktop. Tim also implemented the Resource Construction Set as well as much of DOODLE.

Tim has a master's degree from Michigan State University. He is the author of ST PROFESSIONAL GEM, a regular GEM programming column that you can read on ANTIC ONLINE, a CompuServe information service. As this START went to press, Tim had just accepted a position with a major Silicon Valley firm. He plans to spend his free time hiking throughout the Santa Cruz mountains.

Mark Skapinker is Director of Product Development at the Canadian software firm, Batteries Included. He is author of Thunder! and coauthor of the Isgur Portfolio System. Mark was raised in Africa, where he studied psychology at Witwatersrand University in Johannesburg. He later emigrated to Israel, where he lived in a kibbutz for a year while doing postgraduate work in computer science.

Mark's first computer job was developing software for CP/M machines with NISN Information Services in Tel Aviv. There, he ▸

COPY II ST™

BACKUP PROTECTED SOFTWARE FAST.

From the team who brought you COPY II PLUS (Apple), COPY II PC (IBM) and COPY II MAC (Macintosh) comes a revolutionary new copy program for the Atari 520 and 1040 ST computers.

- Copies many protected programs—automatically. (We update COPY II ST regularly to handle new protections; you as a registered owner may update at any time for $15 plus $3 s/h.)
- Supports single and double sided drives.
- Includes both a fast sector-based copier and a true bit copy mode for protected disks.

Requires an Atari 520 or 1040 ST computer with one or two drives.

Call 503/244-5782, M—F, 8—5:30 (West Coast time) with your VISA MC in hand. Or send a check for $39.95 U.S. plus $3 s/h, $8 overseas.

$39.95

Central Point Software, Inc.
9700 S.W. Capitol Hwy #100
Portland, OR 97219

Central Point Software

CIRCLE 010 ON READERS SERVICE CARD

Backup utilities also available for the IBM, Apple II, Macintosh and Commodore 64.
This product is provided for the purpose of enabling you to make archival copies only

AUTHORS

learned by necessity to program microcomputers to accomplish tasks done elsewhere by minicomputers. Mark lives in Toronto, Ontario with his wife, Hazel, and daughter, Lisa.

David Small has contributed a wealth of technical articles to various computer magazines, including Creative Computing, Antic, and the first issue of START (Voodoo Computing). He has three books, including "Guidebook for Winning Adventures" (Baen Enterprises, N.Y., NY) which he coauthored with his wife, Sandy. Their fourth book is expected out in January 1987.

David discovered how to make the Atari ST into a Macintosh (much to Apple's chagrin) and hopes to get the "MacCartridge" to market soon. He has worked for several computer companies and is now a consultant, freelance writer, and busy father of two children.

Xanth Park is not a person, but a collection of talented programmers. Situated in downtown Seattle, XANTH is the largest ST retailer in the state of Washington. It began—and still partly exists—as the XANTH SST (Seattle ST) User's Group. A little over a year ago, the group began selling Atari 130XE's and have since become a full computer store.

The XANTH programmers, having evidently tired of wowing the ST community with graphic demonstrations, are currently putting the finishing touches on a multi-computer, point-of-view, 3D maze game. Although they cherish anonymity, they welcome correspondence:

XANTH Computer Systems
600 First Avenue
Seattle, WA 98104
BBS: (206) 682-8039 ■

RACK YOUR BRAINS

ST-POOL™

Available for the Atari 520ST or 1040ST; Color monitor required.

To order: Contact your Atari ST Dealer or send $34.95 plus $3.50 for shipping and handling to:

Shelbourne Software
7221 Rising Sun Ave.
Suite 191
Philadelphia, PA 19111
(215) 725-5644

(PA residents add 6% sales tax. Visa and MC accepted.)

Atari 520ST & 1040ST registered trademarks of Atari Corp. ST-Pool is a registered trademark of Shelbourne Software.

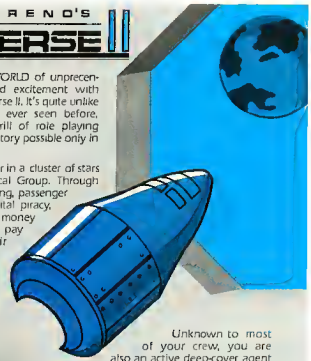
Dealer inquiries accepted

CIRCLE 111 ON READER SERVICE CARD

OMNITREND'S

UNIVERSE II

ENTER A NEW WORLD of unprecendented depth and excitement with Omnitrend's Universe II. It's quite unlike any game you've ever seen before, combining the thrill of role playing with the depth of story possible only in text adventures



You are a free trader in a cluster of stars known as the Local Group. Through sharp trading, mining, passenger transport, and orbital piracy, you'll try to earn money to fuel your ship, pay your crew, repair parts, acquire advanced technology, and perhaps buy a new spacecraft.

Unknown to most of your crew, you are also an active deep-cover agent for one of the governments in the Local Group. As the interstellar situation worsens, you'll be called upon to carry out covert operations deep within enemy territory

Universe II is available for the Atari ST, Apple II, Macintosh, and IBM computers Price: $69.95. Also from Omnitrend: Universe I for the Atari XL/XE, Apple II, and IBM computers Price $59.95.

To order contact your local dealer or call Omnitrend at (203) 658-6917
P.O. Box 3, West Simsbury, CT 06092

OMNITREND SOFTWARE

CIRCLE 080 ON READER SERVICE CARD

Improve your 520 ST's performance

with the ThoughtSpace TS-1A one megabyte expansion board

The TS-1A gives you greater speed and power. Bigger documents and applications. More desk accessories. Larger RAM disks. Full 1040 ST software compatibility.

Whether you have it installed or do it yourself, you get a complete, tested board with a utilities diskette and detailed instructions.

The TS-1A is of the highest quality, comes with a one year guarantee, and is simple to install.

Call or write for literature, dealer list, and ordering information.

only $150 (+$15 installation)

ThoughtSpace Development

2450 Warring St., No. 21 Berkeley, CA 94704
(415) 845-1415
Dealer inquiries invited!

CIRCLE 122 ON READER SERVICE CARD

ATARI ST

ACTION PAK™

4 Atari ST™ Programs
1 Low Price!

39.95

DISK LABELER

Custom design your own unique 3.5 inch disk labels.

- Create professional-looking labels using three windows on screen. Fully mouse driven.
- Up to 38* files on each label, alphabetically, with monochrome monitor. Up to 18 files with color monitor.
- 35 borders and 4 type styles*.
- Includes 50 pin feed disk labels for 3.5 inch disks.
- Uses ST's full graphics capabilities*.

BANNER MAKER

Express yourself! Create original 1 to 4 line banners up to 72 characters in length.

- Includes 5 fonts. Custom design your own with Degas** or use any fonts compatible with Degas**.
- Left, right or center justification.
- Does *not* require "Action Pak (4 in 1)"** to print banners. Banner files can be given on disk or sent by modem.

TYPEWRITE

Great for addressing envelopes, filling out forms or writing short memos.

- Turns your ST into a line-at-a-time typewriter.
- Always handy! Resides in RAM as a desk-top accessory or runs as a stand alone program.

SYNFILE** CONVERTER

*A real work-saver! Convert your 8-bit Synfile** files fast without having to reenter data.*

- Creates ST files compatible with Hippo Simple**, Zoomracks**, DB Master One**, H & D Base**, DB Man**, dbOne**, or Regent Base**.
- Easy to use! Completely self-prompting. Just answer the simple questions.
- Requires modem or null modem.

Available at your local ST retailer. If you are unable to find it, then send coupon to: Action Software — 69 Clementina, Dept. SA, San Francisco, CA 94105. Please include $3.00 shipping & handling. Calif. residents please add 6.5% sales tax.

☐ Check enclosed Bill my credit card: ☐ VISA ☐ MC

Name _____

Address _____

City _____ State _____ Zip _____

Credit card no. _____ Exp. Date _____

Signature _____

We are looking for quality ST programs to publish. Call (415) 974-6638.

*Requires Epson** or Epson** compatible graphics printer. Other non-Epson** compatibles will print up to 18 file names, but may not accept graphics.

**"Action Software, Atari Corp., Batteries Included, Synapse Software, Quickview Systems, Mirage Concepts, Versasoft Corp., Oxxi, Inc., Regent Software and Epson Corp. respectively.

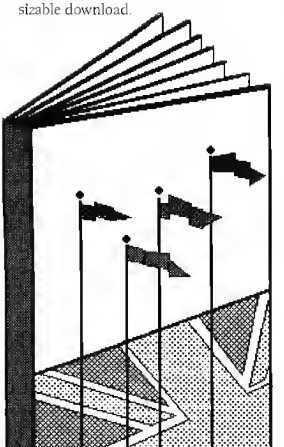
CIRCLE 003 ON READER SERVICE CARD

RESOURCES

START supports the efforts of the whole Atari ST community. In our Resources section we will help you find sources of information and assistance to make your ST computing more rewarding. If you want START to publish your message, address your request to START Resources, c/o Antic Publishing, 524 Second Street, S.F., CA 94107. We'll do what we can, but no promises.

INFOTREE

Modem owners should know that Atari has a Bulletin Board System, (408) 745-5308. Some other BBS numbers for ST owners are available from BUG, the Boise Users Group. Their BBS number is (208) 383-9547. ST boards are intermixed with Atari 8-bit boards, but they are identifed as ST. Be prepared for a sizable download.



JIM WARNER

DOWN TO BUSINESS

The ST is a capable business machine, and that's the angle ST Business takes. This new publication bills itself as The Independent Publication for Atari Small Business Computer Owners. For more info write Don and Carole Terp, Editors, at 5140 Appletree Drive, Roanoke, VA 24019

FESTS and FAIRS

Atarifests and Atari Fairs are popping up all over. Usually these are annual shindigs sponsored by a coalition of users groups in a large geographic area. For example, five groups from northern California will meet at the San Jose Convention center on Sept. 20 and 21. An Atari fair in Portland, Oregon is slated for Oct. 17-19. If you know of special Atari events coming up, give us plenty of notice and we'll try to announce them.

WHAT'S IN A NAME?

Just as ACE (meaning Atari Computer Enthusiasts) became a common suffix for many Atari 8-bit users groups (MACE, SPACE, PACE, etc.), STers may find STING useful (ST INformation Group). The first STING we know of serves the southern Illinois area: contact David Stambaugh, 109 Florida St., Washington IL 61571. Thus, the COAST users group (California Original Atari ST etc.) could become COASTING. Howls of rage can be registered with Chuck Thorp, COAST president, 1015 Esther Drive, Pleasant Hill, CA 94523. Their newsletter is called FirST, and costs $2 per month for nonmembers.

Another group we've heard of calls themselves Jbug, which somehow translates to Atari ST Users Group of the Boston Computer Society. Call (617) 296-8286 for information.

HANDBOOK

The "Atari ST User's Handbook," by Gilbert Held, is available for $9.95 from Weber Systems Inc., 8437 Mayfield Road, Chesterland, OH 44026. It's 160 indexed pages are well illustrated to show the new user how to set up the system and use the GEM Desktop. It includes a chapter on programming in Logo and one on data communications. The book mostly covers material already in the owner's manual, but does it better, if that's what you need.

OVER THERE

Popularity of the Atari ST in England is evidenced by ST User, a monthly magazine from Dufose Publishing, Ltd., 43 South Street, Chichester, West Sussex, PO19 1DS, England. Americans can subscribe for 36 British pounds per year. The content is heavy on reviews, with some tutorials and type-in programs.

OUR OWN HORN

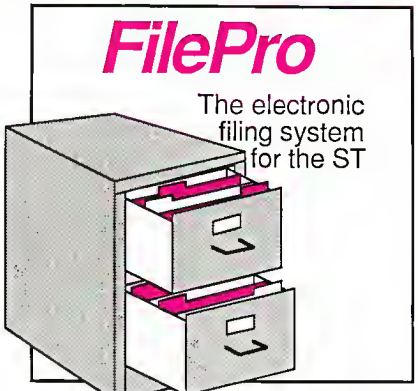
ANTIC ONLINE, an electronic information service of Antic Publishing, is carried by CompuServe. Our refurbished menu offers ST-specific news, reviews, and product information. Simply log onto CompuServe, type GO ANTIC and look for the ST Section. Antic also sells ST software through the mail; see our offerings elsewhere in this issue.

THE BOYS ARE IN CHARGE

Local computer tabloids are doing well in several markets. MicroTimes, California's (free) Computer Magazine, featured the STs and interviews with Sam and Len Tramiel in its July, 1986, issue. Copies may still be available from BAM Publications, 5951 Canning St., Oakland, CA 94609.

DYNAMITE ST SOFTWARE

that won't blow your budget



PaintPro is a friendly, yet powerful design and painting package for drawing graphic and artistic pictures. This GEM-based package supports up to **three** active windows—cut & paste from one window to another. Complete toolkit of functions: free-form sketching, lines, circles, ellipses, boxes, fill, copy, move, spray, zoom, undo, help and extensive text capabilities. Import "foreign" pictures for enhancement using **Paintpro**'s double-sized picture format and send hardcopy to most popular dot-matrix printers. **PaintPro** works with either monochrome or color systems. **$49.95**

FilePro is a simple-to-use but powerful data management system. Drop-down menus to quickly define your file and enter information. **FilePro** has many features that make it unique and flexible: store data in different type styles; create subsets of a file; change file definition and format; includes a RAM disk for high-speed operation. Has a fast search and sort capabilities, handle records up to 64,000 characters, 15 significant digits for values, access up to 4 files simutaneously, index up to 20 different fields/file and has complete report capabilities. **$49.95**

Other sensational ST software available:

ST Forth/MT - Powerful, full-featured, multi-tasking Forth. 32-bit implementation based on Forth83 standard. Full screen editor, monitor, macro assembler. 1500+ word library. **$49.95**

ST Text Designer - Ideal package for page layout. Accepts prepared text files from TextPro, other ASCII wordprocessors. Perform block operations—copy, move, columns. **$49.95**

ST AssemPro - Professional developer's package. Editor, interactive two- pass assembler with error locator, online help, monitor- debugger, disassembler, 68020 simulator. **$59.95**

ST TextPro - Wordprocessor with professional features and easy-to-use! Full screen editing. sideways printing; multi-column output; automatic index and table of contents. **$49.95**

PCBoard Designer - Interactive, computer-aided design package for pc boards layouts. Auto-routing, pinout & component list, Two-sided boards. 45° & 90° traces. **$395.00**

Call **now** for the name of your nearest dealer. Or order directly with your MC, VISA, or Amex card. Add $4.00 per order for postage and handling. Foreign add $10.00 per item. **Other software and books coming soon.** Call or write for your **free** catalog. Dealer inquiries welcome-1400 dealers nationwide.

Abacus Software

Phone (616) 241-5510 - Telex 709-101

P.O. Box 7219 Dept. TA Grand Rapids MI 49510

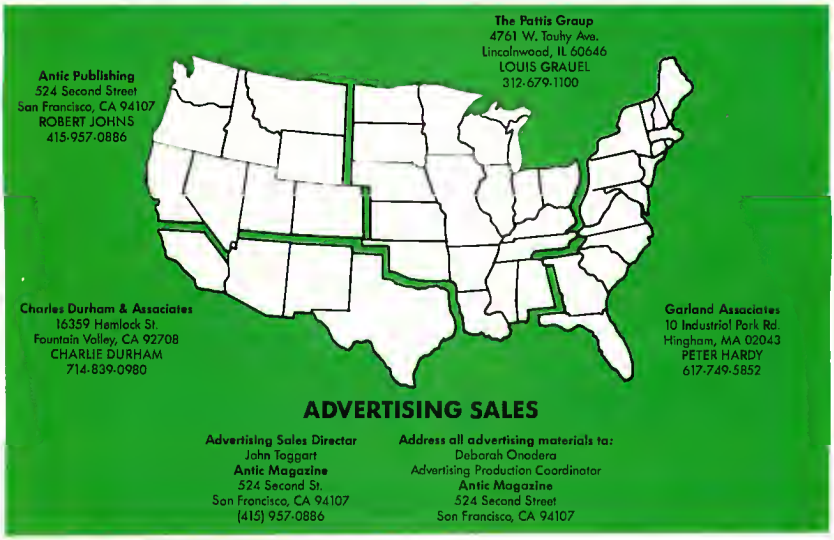
CIRCLE 001 ON READERS SERVICE CARD

ADVERTISERS LIST

Be sure to mention ANTIC when contacting these advertisers—all of whom support the Atari ST Computer.

	READER SERVICE NO.	PAGE NO.
ABACUS SOFTWARE	001	121
ABBY'S	002	95
ACTION SOFTWARE	003	8,119
ACTIVISION	067	123
ADVANCED COMPUTER SERVICES	004	17
ANTIC		18
ATARI	005	7,9,11
BATTERIES, INC.	006	45,124
BAY VIEW SOFTWARE	007	89
BECKMEYER	008	115
BLACK PATCH	009	35
CENTRAL POINT	010	117
COMPUSERVE	011	3
COMPUTER CREATIONS	012	49
COMPUTER PALACE	013	51
CREATIVE SOLUTIONS	014	53
DATASOFT	015	10
DISCLONE	016	76
DIVERSE DATA	018	76
DUPLICATION TECHNOLOGY	019	31
ELECTRONIC ONE	020	8
ELF SOFTWARE	021	74
GEMINI ENTERPRISES	030	74
IB COMPUTER	040	61
LYCO COMPUTER	050	80
MEGAMAX	062	74
MEGASOFT	063	114
METACOMCO	066	89
MICHTRON	064	47,108
MICROPROSE	065	21
MICROTYME	069	76
NAVARONE	070	79
OMNITREND	080	119
PALO ALTO SHIPPING	090	79
PROCO PRODUCTIONS	091	119
PROGRESSIVE COMPUTER APPLICATIONS	092	2
REGENT SOFTWARE	100	106
S & S WHOLESALE	110	115
SHELBOURNE SOFTWARE	111	118
SOFTWARE DISCOUNTERS	112	89
SOUTHERN TECHNOLOGY	113	8
STATIC ENGINEERING	114	88
ST PLUS	115	85
TDI	120	119
TERRIFIC PERIPHERHALS	121	61
THOUGHT SPACE DEVELOPMENT	122	119
TIMEWORKS	123	4
VIP MARKETING	017	81
XLENT	130	20

This list is provided as a courtesy to our advertisers. ANTIC does not guarantee accuracy or comprehensiveness.



ADVERTISING SALES

Advertising Sales Director
John Taggart
Antic Magazine
524 Second St.
San Francisco, CA 94107
(415) 957-0886

Address all advertising materials to:
Deborah Onodera
Advertising Production Coordinator
Antic Magazine
524 Second Street
San Francisco, CA 94107

FREE TRIP TO RUSSIA

And you don't even have to leave your room.

The Russians have *The Doomsday Papers*™ locked deep in a Siberian stronghold. With them, they can bring the world to its knees.

The U.S. government needs your help. Using their satellite you can get into the complex with your computer. All you have to do is locate the combinations to the safe, find and open it, and get the documents out.

Sounds simple enough. Unfortunately, it's not that easy. There are video cameras and monitors to be avoided. Guard patrols. And something called…The Annihilator. Plus, your only map is the one you've got to make while not being discovered by all of the above.

Your reputation got you into this mess. Your hacking skill is the only thing that can get you out. Bon Voyage.

For more information and the dealer nearest you call 800/227-9759 (in California, call 415/960-0410) weekdays 10:00 a.m. to 4:00 p.m. Pacific time.

Hacker II: The Doomsday Papers designed by Steve Cartwright, who brought you the award winning challenge of Hacker.™

Hacker II for Commodore 64/128, Apple II, IBM PC/PCjr and Tandy 1000, Amiga, Atari ST and Macintosh computers.

ACTIVISION
ENTERTAINMENT SOFTWARE

Commodore 64, 128 and Amiga are trademarks of Commodore Electronics Limited. IBM is a trademark of International Business Machines Corporation. Tandy is a trademark of Tandy Corporation. Atari and ST are trademarks of Atari Corp. Apple and Macintosh are trademarks of Apple Computer. Activision is the registered trademark of Activision, Inc. © 1986 Activision, Inc.

Advertisement

August 1986

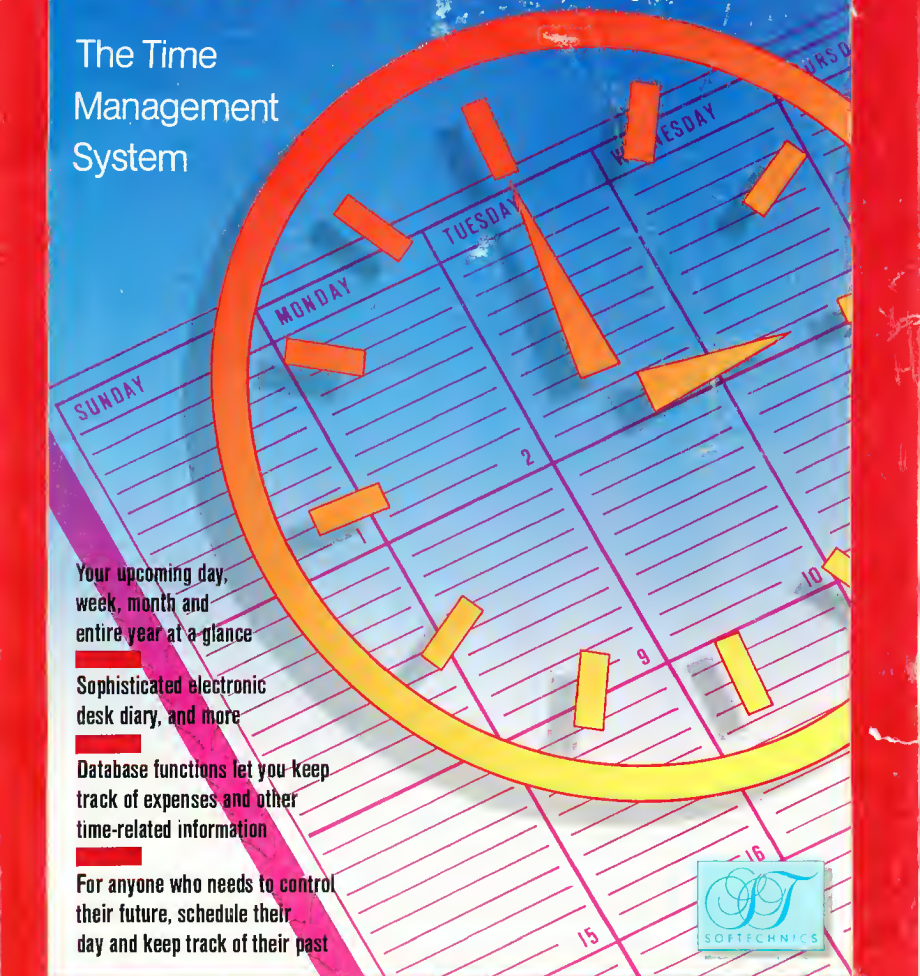
TiME LiNK

$49.95*

BATTERIES INCLUDED

For the MAC and ATARI ST

The Time Management System



Your upcoming day, week, month and entire year at a glance

Sophisticated electronic desk diary, and more

Database functions let you keep track of expenses and other time-related information

For anyone who needs to control their future, schedule their day and keep track of their past

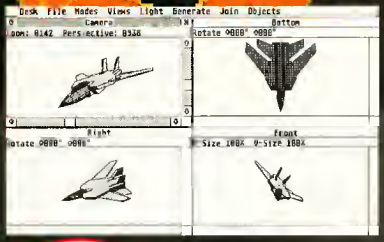
SOFTECHNICS

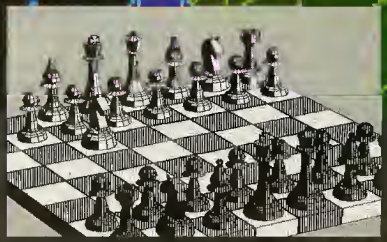
30 Mural Street, Richmond Hill, Ontario L4B 1B5 Canada (416) 881-9941 Telex: 06-986-266 17875 Sky Park North, Suite P, Irvine, California, USA 92714 (416) 881-9816 Telex: 509-139 If you can't find TIME LINK at your local retailer, you can order it direct from us for $49.95* (plus $5.00* for shipping and handling). Call 1-800-387-5707 (U.S. only). Update your disk for only $10.00* with the return of your original disk. *Suggested U.S. list price. All prices shown are in U.S. funds. WRITE TO US FOR OUR FULL COLOUR CATALOG OF SOFTWARE FOR ATARI, ATARI ST, APPLE, MACINTOSH, COMMODORE, AMIGA, AND IBM SYSTEMS. ©1986 BATTERIES INCLUDED. APPLE, ATARI, COMMODORE AND IBM ARE REGISTERED TRADEMARKS RESPECTIVELY OF APPLE COMPUTERS INC., ATARI INC.,, COMMODORE BUSINESS MACHINES INC., AND IBM BUSINESS MACHINES INC.

Advertisement

THE CATALOG™

QUALITY ST SOFTWARE





BEST SELLER

The author of DEGAS™ enters a new dimension.

Tom Hudson's CAD-3D™

A full, three-dimensional modeling station for your ST. Solid modeling, with shading from three adjustable light sources, plus wireframe and hidden surface dimensional sculpting—*you name it.* All your 3-D graphics fantasies have been poured into the power and speed of the ST's 68000. So step on the gas and take off!

From starships to solar systems, skyscrapers to your front stoop. Create and animate views of your own 3-D objects. And no keyboarding here—CAD-3D is entirely GEM-based. It's got to be seen to be believed.

DESIGNERS: You now have power at your fingertips that was previously available only on systems costing $30,000 or more. *Visualize ideas instantly*—speeds up design process.

DOODLERS: Easy to learn, easy to use. *Create the ultimate in dimensional scenes.* Then detail them with any ST paint program.

CAD-3D features include:

- ☐ **Works with monochrome or color monitor. Monochrome mode gives 16 levels of shading, color gives choice of 1 color/14 shades or 2 colors/7 shades each.**
- ☐ **View objects in see-thru or hidden wireframe form or *solid form with true light shading.* Change object form with one mouse click.**
- ☐ **Three independent user-defined light sources plus ambient lighting.**
- ☐ **Built-in 3D primitives (cubes, spheres, torolds, wedges).**

continued on next page

- ☐ *Automatic* JOINER creates more complex objects:
  - ADD objects together
  - SUBTRACT one object from another
  - AND objects (retains only the intersection)
  - STAMP one object on another
- ☐ Create radially symmetrical objects or expand 2D shapes into 3D shapes with one mouse click.
- ☐ *Animation capability*—"record" view sequences for viewing with separate display program (included).
- ☐ Spin and Extrude functions allow even the novice 3D experimenter to create a wide variety of beautiful shapes (includes 'Snap To' grid).
- ☐ Independent scaling and rotation of objects or groups of objects.
- ☐ "Camera View" includes zoom lens and variable perspective.
- ☐ GEM user interface allows use of four views at once, or one large view. Four-view mode is user-definable.
- ☐ Save completed images in DEGAS, Neochrome or COLR Object Editor format.
- ☐ Print your objects with a graphics printer.
- ☐ Detailed illustrated user manual.
- ☐ Sample complex objects included, featuring Ian Chadwick's *STONEHENGE* (circa 1100 BC).

Requires TOS in ROM
ST0214 **$49.95**

DEGAS™ Batteries Included

INCREASE YOUR DESIGNING POWER WITH CAD-3D ACCESSORY DISKS!

FONTS, PRIMITIVES, HINTS AND TIPS: NEW!

- ☐ Create logos, signs, 3D letterheads and more with 3D fonts. Includes upper and lower case alphabet, numbers, and symbols.
- ☐ Useful 3D 'primitives'—these building blocks speed up the design of large-scale models.
- ☐ Sample models and a file of Tom Hudson and Ian Chadwick's hints and tips on how to make complete digital scenes with CAD-3D.

AVAILABLE OCTOBER 1ST
ST0224 **$24.95**

See pg. ST-16 for Tom's CAD-3D Collection I. Only $12.00!

PLOTTER AND PRINTER DRIVERS: NEW!

- ☐ Plot your 3D drawings on a high-resolution pen plotter for a truly professional look! Supports Hewlett-Packard (and HP compatible) plotters. Also includes custom printer drivers for all popular dot-matrix printers.

AVAILABLE OCTOBER 1ST
ST0225 **$24.95**

A Full Function Relational DataBase

REGENT BASE™ NEW!

by Regent Software

You don't need to be a dBase III programmer to use this database. Perfect for the small business owner or for personal productivity and it's GEM-based.

REGENT BASE's procedural language makes it a natural for handling any small business need. Modules are available for Invoicing, Accounts Receivable, Checkbook Balancing, General Ledger, and more.

This relational data base is written specifically for the Atari ST. Don't settle for simple clones of IBM products. **REGENT BASE** is easy to use and state-of-the-art!

Unique to **REGENT BASE** is the use of B+Tree Indexes. Multiple fields within a **REGENT BASE** table may be indexed for quicker access. Use the Bridge program to move and merge other database information. Sort, file, mail-merge, select up-graded word processing functions, unlimited records and indexes per table.

REGENT BASE runs efficiently on hard or floppy drive systems and supports 15 printer types.

Recommended—
1 Megabyte RAM
TH9003 **$99.95**

TOLL FREE
Credit card holders, call toll-free, 24 hours-a-day
VISA 800-443-0100, ext. 133 (Continental U.S. and Hawaii)
ORDERS ONLY!

Pure telecommunications power...Realized.

BEST SELLER

FLASH™

Version 1.1

by Joe Chiazzese and Alan Page

"FLASH is my #1 choice on the Atari ST."

*– Ron Luks, Founder of CompuServe's SIG*Atari and Atari Developers Forum.*

Want to take telecommunications to the outer limits of your imagination? All you need is **FLASH** and a modem to get you there. We wanted the *ultimate* in telecommunications for the Atari ST. And it had to be affordable.

We couldn't wait. We built it. ...Plus, it's GEM-powered.

Now you can have the automation features and VT100 terminal emulation of Crosstalk™ on the IBM and the intuitive mouse/window interface of Smartcom™ on the Mac. Plus, the built-in GEM-based word processor is integrated with the capture buffer—to give you a scrollable, editable history of every online session.

You can have all of it *now*, for under $40. Plus, we'll give you $15 of CompuServe connect time *FREE!* (Effectively bringing the cost of the software down to under $25!)

FLASH features include:

- ☐ Built-in GEM mini *word processor* (features block move, undo, search, merge files, and more)
- ☐ Review your online session. Scroll through it at high speed—edit it, print it, transmit all or part of it, save all or part of it to disk. Huge capture buffer automatically adjusts to fill available RAM.
- ☐ Create "script" files that automate *any* operation—log-ons, file transfers, "broadcasting," email collection, etc. (69 powerful commands)
- ☐ Execute a complete online session with *one* mouse-click from the desktop.
- ☐ ANSI VT100 and VT52 cursor and keypad editing emulation (full 24 line ×80 character smooth scrolling display).
- ☐ CompuServe Vidtex high-resolution graphics terminal emulation. Save Vidtex graphics as DEGAS files and use them as *clip art!*
- ☐ Supports Xmodem (CRC & Checksum), DC2/DC4, and ASCII TEXT file transfer protocols.
- ☐ Ultra-reliable. **FLASH** features *bullet-proof* file transfers to and from CompuServe (and other packet-network online services) at speeds up to 9600 baud.
- ☐ Extensive DOS functions at your fingertips. Two clocks: built-in real-time system clock and elapsed timer.
- ☐ Unlimited banks of twenty programmable function keys, which can be linked to script programs on disk.
- ☐ Translation tables can independently filter any incoming or outgoing characters.
- ☐ Detailed 50-page user manual by Ian Chadwick, including complete reference guide.

ST0220 — Crosstalk™ Microstuf — **$39.95** — Smartcom™ Hayes

FREE! Try FLASH now and get $15.00 of CompuServe access time, FREE.

Order now and you'll receive a CompuServe Intro-Pak; your free introductory subscription to the CompuServe Information Service with a 46-page CompuServe mini-manual. Plus $15.00 worth of CompuServe access time, free.

This offer is valid for everybody, including current CompuServe subscribers.

FREE OFFER! Here's what you have access to on CompuServe as soon as you receive your copy of **FLASH:**

■ **ATARI 16-BIT FORUM:** The best ST PD software, support, and gossip.

■ **ATARI DEVELOPERS FORUM:** Official Atari support, and talk of the trade.

■ **ANTIC ONLINE:** Hot news, special articles never released in print.

■ **VIDTEX GRAPHICS:** Hundreds of pictures. Everything from Hollywood stars to the FBI's most wanted. Outrageous stuff—*the ultimate clip art library!*

KERMIT SERVER & REMOTE CONTROL ACCESSORY

NEW!

Adds the Kermit file transfer protocol to **FLASH.** Plus, you can now call your computer from anywhere in the world and have control over: disk directory, send/receive Xmodem, send/receive Kermit, Kermit server, and upload/download text files. Your system is secured with real password protection. *Perfect for sales, business, and school.* Now you can use your ST from your office or while you're on the road.

AVAILABLE OCTOBER 1ST

ST0226 — **$24.95**

Requires FLASH

INTEGRATED FINANCIAL SOFTWARE FROM BATTERIES INCLUDED

THE ISGUR PORTFOLIO™

ANTIC AWARD

THE ISGUR PORTFOLIO is three programs in one: Portfolio manager, telecommunications, and calendar/memo pad. Receive Dow Jones stock quotes, CompuServe, The Source, and InfoGlobe financial information, automatically. Consolidate multiple stock portfolios. Hands on managment of stocks, bonds, options, commodities, mutual funds—your key to financial success.

THE ISGUR PORTFOLIO is PC magazine's "EDITOR's Choice." The integrated GEM-based system provides intuitive access to your data and unlimited versatility. A complete financial management package for the professional and individual investor.

A real value for your investment. Plus, **THE ISGUR PORTFOLIO** includes **I•S TALK**. Invest in your future...today.

TH9009 **$199.95**

D.E.G.A.S.™

BEST SELLER

by Tom Hudson

"A GRAPHICS MASTERPIECE..." —Analog Magazine

DEGAS features all the artistic tools that you may need including:

- ☐ Unique dual-screen GEM interface makes **DEGAS** intuitive and fun to use.
- ☐ Works in any resolution.
- ☐ Numerous functions to create and draw lines, rays, circles, boxes, or frames. Create perfectly straight lines or beautiful circles automatically.

PLUS ADVANCED GRAPHIC DESIGN FEATURES:

- ☐ An airbrush effect that lets you control the "paint flow."
- ☐ Automatic "Fill" function. Fill any outline instantly.
- ☐ Create "Fill" patterns.
- ☐ Instant Mirror Image, in any direction.
- ☐ Automatic Shadow or Border, you control the width and the angle.
- ☐ Zoom window lets you work with fine-detail designs.

PLUS NOW YOU CAN INTEGRATE WORDS WITH YOUR VISUALS.

- ☐ Use the Text feature to add words to your art.
- ☐ Choose the character weight and size from various fonts, or create your own with the font editor.

Now you have the power to reach the third dimension with your **DEGAS** pictures using the powerful **CAD 3-D** system on page ST-1.

TH9008 **$39.95**

A spelling checker that is lightening fast!

THUNDER™

THUNDER gives you a powerful spelling checker accessible from within your favorite application program. Your accuracy will skyrocket, so the time spent proofing will plummet.

It's so simple. Here's how it works. Example: you're happily writing away and you make a "mstake" and BEEP (**THUNDER** picks up the error with lightning-fast speed). Now **THUNDER** will display a list of similarly spelled words (stake, mistake, etc.)

Compatible with Paperclip Elite, Homepak, BTS, other Batteries Included products, 1st Word, GST-Edit, DB Master One, as well as FLASH on page ST-3, A-CALC on page ST-5, and many more GEM software titles.

TH9011 **$39.95**

TIMELINK™

Take A Second For Time Management

TIMELINK is a great GEM-based scheduling and time-keeping tool for home and business. Your day, week, month and year appointments are only a mouse-click away. Useful for messages, reminders and much more. There are many incredible uses for the handy elegant, time-saving tool.

TH9010 **$49.95**

TOLL FREE

Credit card holders, call toll-free, 24 hours-a-day
800-443-0100, ext. 133 (Continental U.S. and Hawaii)

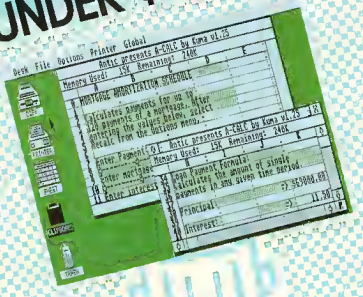
ORDERS ONLY!

A-CALC™ by Kuma

BEST SELLER

A FULLY GEM-BASED SPREADSHEET FOR UNDER $60.

The first mouse-driven spreadsheet for the Atari ST computer system. A-CALC's GEM interface is exceptional. For example, load a spreadsheet simply by dragging the disk icon over the sheet icon. Imagine how *easy* it is to merge a file from disk by dragging the disk icon into the appropriate cell.



Version 1.50

by Kuma Systems, Ltd., U.K.

A-CALC's dozens of features include:

- ☐ Primarily mouse/icon driven. Keyboard or ten-key calculation entry.
- ☐ Math functions include all standard algebraic and trignometric functions for performing financial and statistical analysis.
- ☐ Up to five windows can be open at once.
- ☐ Spreadsheet capacity of 256 columns × 512 rows (128,000 cells).
- ☐ Cells, blocks, rows, and columns can be copied, moved, saved, printed or deleted with the mouse.
- ☐ Special WIDE-VIEW function permits half-size text display to let the user see more of the sheet at one time, in color or monochrome.
- ☐ SEARCH for the location of any text string in the sheet. GOTO command jumps to any cell.
- ☐ Clipboard for temporary storage of block, row or column of cells which can be transferred to another spreadsheet.
- ☐ Powerful printer formatting commands. Comes with an easy-to-use GEM-based printer installation program for any parallel or serial printer. Customize your spreadsheet with the following:
  - headers/footers
  - horizontal/vertical dividers
  - selectable form feeds
  - selectable page break
  - send printer-specific control codes
  - character translation
  - loading and saving printer configuration files
- ☐ Includes on-disk spreadsheet examples.
- ☐ Worksheets are compatible with all popular word processing software. Plus, A-CALC is .DIF compatible with other spreadsheets.
- ☐ "Sparse matrix" spreadsheet design *maximizes available memory.* Create extra-large spreadsheets with RAM to spare.
- ☐ Cells can be protected from tampering —locally or globally.
- ☐ First-ever use of "Dynamic icons" makes operation totally intuitive.
- ☐ Includes illustrated manual.

ST0212 **$59.95**

A-CHART, THE solution for presentation-quality graphics, will be available in early October. Check for availability with your local retailer.

ARTIFICIAL INTELLIGENCE

"That's right! Expert Systems, the artificial intelligence tool for business, industry, and problem-solving. The latest in knowledge engineering."

EXPERT OPINION™

Fifth-generation. Artificial intelligence. Now, with **EXPERT OPINION**, you can be creative in this growing new field...without having to learn a complicated language like Prolog or Lisp.

Be The First...

EXPERT OPINION is the *first* general-purpose expert system creation package for the Atari ST. Expert systems are collections of natural language rules that you create, database-style. Your "knowledge bases" can be about any subject. **EXPERT OPINION** will give you new insight and advice about your expertise. And, because **EXPERT OPINION** uses the GEM interface, the system is remarkably simple to use. Currently, expert systems are used in the fields of medicine, computer science, economics, and geology.

YOU CAN BUILD YOUR OWN KNOWLEDGE BASE WITH THE INCLUDED MANUAL AND SUGGESTED SUPPLEMENTARY READING MATERIAL.

Mainframe Power In Your ST...

Once you've built your knowledge base, **EXPERT OPINION** is easy-to-use because it's based on a powerful natural language interface, so you can give your input—and get your answers in plain English. Plus, it is the only expert system presently available for micro-computers which can clearly explain why a particular question is being asked (it backtracks up to 10 levels).

EXPERT OPINION's "inference engine" solves problems three ways:

- ☐ **DEDUCTION**—Use this mode if you have some initial data about the problem (also known as Forward Chaining).
- ☐ **VERIFICATION OF A HYPOTHESIS**—Your computer asks you questions about your hypothesis (also known as Backward Chaining).
- ☐ **EXPERTISE** (totally new technique)—A combination of the above, for situations where you have no initial data about the problem, and no hypothesis (Mixed Chaining Mode).

EXPERT OPINION features:

- ☐ Dictionary linked to each knowledge base.
- ☐ Unlimited number of knowledge bases.
- ☐ BaseManager has powerful tools for creating and editing knowledge bases.
- ☐ Illustrated manual includes tutorial and tips by Christopher F. Chabris, Glossary, Bibliography, and more.
- ☐ GEM-driven interface, on-line [Help], and user-friendly command structure make **EXPERT OPINION** simple to use.

FREE BONUS!
Comes with example knowledge bases on disk to get you started.

ST0219 **$99.95**

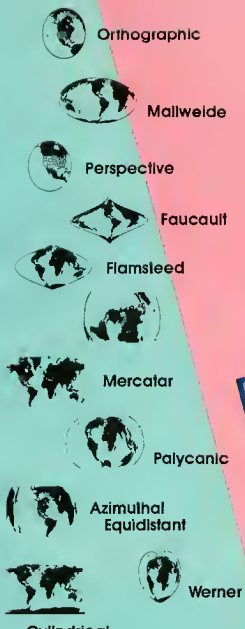
WARNING: This is a sophisticated computer science tool requiring study to use it effectively. We recommend the following books to help you use the program to its fullest:
THE COGNITIVE COMPUTER, Addison Wesley 1984
EXPERT SYSTEMS, Wiley Press 1985
INTRODUCTION TO ARTIFICIAL INTELLIGENCE,
Addison Wesley 1985

THE CATALOG WANTS YOU! PUBLISHING OFFER.

A New Opportunity.

We're looking for people to create useful knowledge bases using EXPERT OPINION. If you're an expert on a subject (any subject), get EXPERT OPINION—write a knowledge base... And if it's very good, and has broad appeal, we'll publish it for you in this catalog. Write to me, Catalog Product Manager, for more details (my address is on the back page). Give me your expert opinions. I'll try to help you publish some of your knowledge bases on a commercial basis.

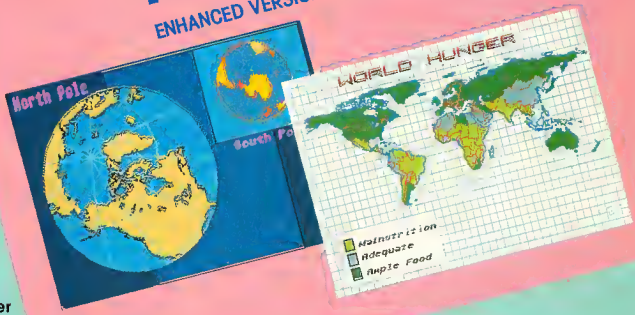
TOLL FREE
Credit card holders, call toll-free, 24 hours-a-day
800-443-0100, ext. 133 (Continental U.S. and Hawaii)
ORDERS ONLY!



Now you can customize MAPS AND LEGENDS to plot any coordinates you want!

MAPS AND LEGENDS™ The Cartographer

ENHANCED VERSION 2.0



by Harry Koons and David Chenette

Put yourself anywhere in the world — even your own hometown. And now view that land mass from any altitude—in full perspective!

UnIimited Versatility

The *new* OVERLAY feature in version 2.0 gives you total flexibility and professional power. Now, *plot your own maps in scale on top of the built-in world map database.* Or use the new DATAMAP disks to add the boundaries of the nations of the world, historical maps, and more.

Tap Your ST's Speed...

Using advanced CAD system algorithms, MAPS and LEGENDS takes less than a minute to do what used to take hours for computers—and weeks by hand. It features ultra-fast drawing of world maps from one of 11 perspectives, three resolutions, up to 16 colors.

ENTIRELY GEM-BASED FEATURES INCLUDE:

- ☐ OVERLAY option for plotting your own maps, flight plans, routes, city locations, etc.
- ☐ BUILT-IN DRAWING TOOLS
  - Multiple Pattern Fills, Pen sizes, Text sizes, Fonts
- ☐ MULTI-FONT DESK ACCESSORY
  - Comes with extra fonts for customized legends
- ☐ AUTO-LOCATE MODE
  - Reads geographic coordinates from map, and distance and bearing from any point on the Earth—just by pointing with your mouse!
- ☐ DEGAS and Neo compatible. Use your maps as basic elements in the most popular ST graphics packages.
- ☐ Works with color and monochrome systems.
- ☐ Built-in topic-based [Help] system.

NOTE: With the new overlay enhancement, MAPS and LEGENDS has become one of the *best* cartographers tools in history. We added this capability to the system after a request for it from a navigational flight officer. He told us that he and the other flight officers at his base were using MAPS and LEGENDS and ST's to draw their flight plans, and that they needed to be able to accurately plot their routes. We liked the idea so much that now anyone can build custom map databases out of simple coordinate lists. And arguably the best thing about version 2.0 is the price...still $34.95.

ST0202 **$34.95**

INTRODUCING DATAMAPS

DATAMAP COLLECTION I: NEW!

NATIONAL POLITICAL BOUNDARIES
Europe, Latin America, S. America, Asia, Africa

PROVINCIAL AND STATE BOUNDARIES
Australia, Canada, China, U.S.S.R. (the SSR's)

HISTORICAL MAPS
Pre World War I Europe, Pre World War II Europe, Original 13 U.S. colonies, and more...

Requires Maps and Legend 2.0
ST0227 **$24.95**

Exciting World War II Submarine Action in the South Pacific!

NEW!

SILENT SERVICE™

By MicroProse

0400 hours, Tuesday, August 12, 1942...BATTLE STATIONS!! BATTLE STATIONS!! Enemy convoy identified on radar!!

SILENT SERVICE, The Submarine Simulation, brings exciting action, great strategy, detailed graphics and an ultra-realistic simulation of World War II U.S. submarine action in the South Pacific.

SILENT SERVICE's outstanding features include: all the critical battle stations, engine room, conning tower and ship's bridge; challenging and realistic combat versus single ships and heavily escorted convoys; and an infinite variety of situations using complete maps and charts for the entire Southwest Pacific and a sophisticated and realistic attack plotting system.

SILENT SERVICE provides a wide selection of historic scenarios. From hit-and-run attacks to patrol missions that bring challenge and fun to both the first time player and the experienced submarine veteran.

As captain, you select a quiet patrol sector in the Marianas Islands, or choose the dangerous waters off the coast of Japan. Is the submerged daylight periscope attack best, or do you charge in on the surface at night using only radar bearings as your guide? These and many more decisions will determine your place among the elite ranks of the **SILENT SERVICE.**

TH9016 **$39.95**

GET ORGANIZED AND HAVE FUN WITH MICHTRON

NEW!

CORNERMAN

Eliminate the desk battlefield and make organizing easy!

What Sidekick did for the IBM, **CORNERMAN** does better for your Atari ST. This utility gives you a host of useful desk-top tools in one simple, neat package. With everything from a built-in clock, notepad, phone book and ASCII table, to a full function calculator, a cluttered desk is a thing of the past. And as a desktop accessory, **CORNERMAN** is available nearly anywhere within GEM.

CORNERMAN doesn't interfere with other programs and comes complete with security. HIRE THE PERFECT RIGHT HAND PERSON TODAY—HIRE **CORNERMAN.**

One megabyte RAM recommended.

TH9013 **$49.95**

CARDS

CARD games for the entire family and serious strategy card player. Beat the dealer in Blackjack, skunk your opponent in Cribbage, play out the aces in Solitaire and win at Klondike and Poker Squares. Incredible graphics.

TH9015 **$39.95**

TIME BANDIT

Now you're organized—so take a break and have some fun.

The arcade adventure takes you to the world of the biggest, fastest, most detailed game ever designed for a home computer. You're the bandit rogue travelling through time and space. The quest always beckons: recover the Great Artifacts and break the Gates of Time! Features: 16 unique arcade lands, 3,000 screens, realistic detail, three text adventures, unique dual-player mode. Uses color monitor.

TH9014 **$39.95**

MichTron

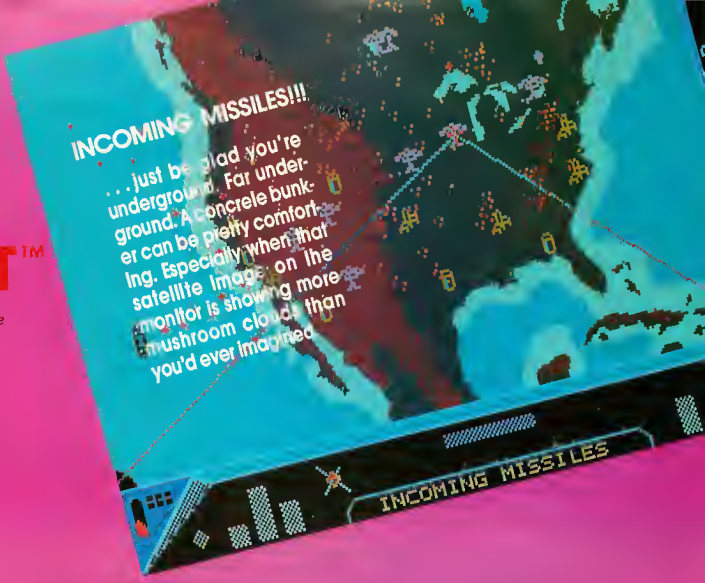
TOLL FREE

Credit card holders, call toll-free, 24 hours-a-day
VISA **800-443-0100, ext. 133** (Continental U.S. and Hawaii)

ORDERS ONLY!

RED ALERT™

Written by Stanley Crane and Daniel Matejka



But you have some options. A few. Strategically-placed Ground-Based Lasers and Antiballistic Missile Silos are the backbone of your defense. Your last space-based Particle Beam Weapon is available, but you must allocate your resources wisely to use it.

Your cities are under attack. It's up to you to try and destroy the Russian ICBM's and MIRV's—plus the missiles launched from nuclear subs. RED ALERT features real-time animation, on-screen help, auto-demo mode, color/monochrome compatibility, three selectable levels of play, PLUS a unique construction set which lets you customize North America with the cities that you want to protect, and the positions of the Ground-Based Lasers and Antiballistic Missile Silos.

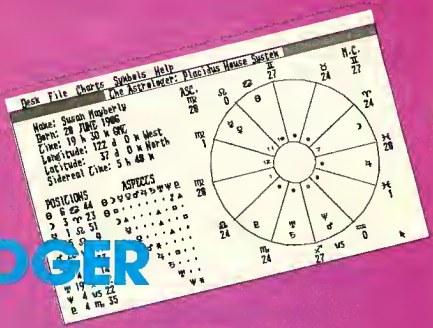
NOTE. RED ALERT is an addicting game, but it's also a thought-provoking version of an SDI (Strategic Defense Initiative) nightmare scenario. (RED ALERT was developed by the authors of DB Master.)

ST0223 **$24.95**

The Perfect Addition to Your Family Album!

STAR STRUCK™ —THE ASTROLOGER

By Harry Koons



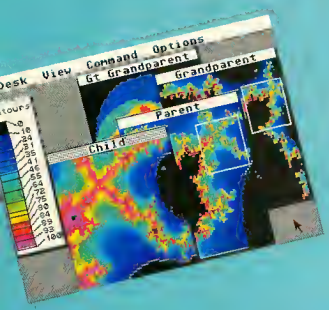
Now your ST can *instantly* generate accurate natal charts for you, your family, and friends—the way an astrologer would do it. Entirely GEM-based, STAR STRUCK creates charts using formulas for the nine most popular house systems used throughout history (Placidus, Morinus, Equal Spaced, Porphory, and five more). On-screen help windows and reference documentation show you exactly what all of the astrological icons mean.

Excellent Graphics, Custom Fonts...
STAR STRUCK includes the Font Loader desk accessory and fonts (gothic, roman, etc.) to give your charts that authentic look. All astrological aspects are displayed in graphic form. U.S. and world maps are built into the program. Locate your birthplace with the mouse or key-in the exact coordinates. Charts can be printed out or saved to disk in DEGAS format. Compatible with color and monochrome monitors.

ST0222 **$24.95**

GSTC APPLICATION FOUNDATION I
FRACTAL FACTORY™

NEW!



Discover how to do graphics programming! Learn how to write a GEM application! Learn by example with the source code for FRACTAL FACTORY. This remarkable disk contains *all* the source code, batch files, linker files, and graphics and math libraries for a unique graphics application written by the GST in-house programmers. FRACTAL FACTORY is the ultimate fractal-maker, using multiple windows, variable zoom, and contouring to display many generations of dazzling fractals at once. Monochrome and Color versions illustrate how to write GEM graphics applications for both types of monitors. The FRACTAL FACTORY source code can be the *foundation* for almost any type of GEM application written in GSTC.

ST0228 **$19.95**

Sixty Eight K Assembler, Editor, Debugger combo for under $35.00!

A-SEKA™ by Kuma

Version 1.6

By Andelos Systems/Kuma, UK

Sometimes you just need to get that code running faster. A high-level language application needs a burst of energy. Or maybe it's arcade action—high end stuff. A-SEKA does it *fast*, because it is all in RAM. All of it: The Assembler, Editor and Monitor/Debugger. Those who know how can create exciting codes mighty fast. And if you're learning Assembly, you won't ever have to wait for your latest attempt to go through the assemble and link process.

A-SEKA assembles source codes at over 30,000 lines per minute! And since it can assemble and link simultaneously, you can *run your code instantly.* Of course, A-SEKA is also a macro assembler and uses standard Motorola mnemonics. But what really sets it apart is its powerful machine language monitor, disassembler and symbolic debugger.

FEATURES:

- ☐ Symbol table access.
- ☐ Arithmetic operations. Input in any base.
- ☐ Disassembles 16 lines at a time.
- ☐ Motorola mnemonics.
- ☐ Single step. Trace.
- ☐ Multiple breakpoints. Memory inspect and modify.
- ☐ Line assembler.
- ☐ Examine registers.

VERSION 1.6 ENHANCEMENTS:

- ☐ Full-screen editor, including cut & paste
- ☐ Expanded GEM AES and VDI libraries
- ☐ Compatibility with other assembler source code
- ☐ Support for GEM subdirectories, additional pseudo-op's for printer and Line A control
- ☐ Step command now skips trap and Line A OS calls

ST0216 **$34.95**

NOTE: We're carrying both the SEKA and GST-ASM assemblers because they are complimentary—each is designed to meet the needs of a different type of programming. SEKA is a low-level assembler that is most useful for programmers who like to work entirely at the machine level or want ultra-fast response time. GST-ASM is really a professional developers tool—its extra GEM features and GEM environment make it the perfect assembler to use with any high-level language.

A-RAM™ by Kuma

By Roddy Pratt, UK

Can your RAMdisk partition any size disk emulator you want?

- ☐ A-RAM can.

Can it work with TOS in ROM?

- ☐ A-RAM can.

Can your RAMdisk accelerate your floppy write speed by turning off the verify mode?

- ☐ A-RAM can.

Can you have multiple RAMdisks present at the same time?

- ☐ You guessed it. A-RAM can.

A RAMdisk is an area of memory set aside as a buffer that responds to most of the available disk commands—only much faster. Everybody needs a great RAMdisk, and A-RAM is powerful, simple and flexible enough for *every* application.

ST0215 **$19.95**

TOLL FREE
Credit card holders, call toll-free, 24 hours-a-day
VISA 800-443-0100, ext. 133 (Continental U.S. and Hawaii)
ORDERS ONLY!

FINALLY! GEM DEVELOPMENT AND LEARNING TOOLS AT A SUPER VALUE!

GSTC™ Compiler

Version 1.3

Here is the compiler that 1ST WORD was written in.

Do you want to write GEM-based programs for the Atari ST? With GSTC you can—without spending hundreds of dollars on expensive compilers. Now you can add windows, dialogs, and all the GEM forms to any program — it's *easy* with GSTC. Use your mouse and pull-down menus to write C programs within a desktop menu-driven "shell" environment. GSTC allows compile-assemble-link and assemble-link operations to be batched, avoiding tedious and error-prone command line entry. And it all fits on one single-sided disk. No excessive disk swaps.

At the heart of the GSTC package is the remarkable *GEM Superstructure Library*. This enables the beginner to write GEM applications software at once, without the complex learning curve associated with GEM AES and VDI. Open a fully-functioning window with one call. GSTC is fast—providing compile and linkage turn-around times speedy enough for the most impatient hacker!

GSTC features include:

- ☐ GEM Text Editor (GST EDIT)
- ☐ Linker (GST LINK)
- ☐ C Compiler
- ☐ GEM "shell"
- ☐ 68000 Assembler
- ☐ GEM Superstructure Library
- ☐ GEM bindings, (Standard Unix, GEM VDI, GEM AES, GEM XBIOS, TOS)
- ☐ Comprehensive printed user manual

Version 1.3 enhancements

- ☐ All 40 GEM XBIOS routines, with documentation
- ☐ All 12 GEM BIOS routines, with documentation
- ☐ Hints and Tips on using pointers to simulate structures
- ☐ FRACTAL FACTORY object code

TH9018 **$79.95**

NOTE: We searched all over the world to find the best introductory C compiler. When we discovered that GSTC was used to write 1ST WORD, we decided that it was just what we were looking for. This compiler is very powerful and remarkably easy to use. But at present, it doesn't have structures or a floating point library. To help you here, our version 1.3 of the GSTC documentation includes information about how to simulate structures with pointers, and the FRACTAL FACTORY disk has a math library on it. If you're writing a program that uses very serious math, you may need to look at Lattice C. But for most applications, GSTC will do the job.

GST-ASM™

A high-level Macro Assembler with an unbeatable combination of price, performance, and features.

Elegance and sophistication... GST-ASM

GST-ASM is a Motorola-compatible 68000 macro assembler designed to work with high-level languages (C, Pascal, Fortran, etc.). Now, assembly language programming is truly accessible to all ST users. And because GST-ASM uses the GST GEM interface and text editor, it's a joy for beginner and advanced programmers alike.

Includes Macro Tools and more!

GST-ASM has high-powered macro facilities, for extremely fast throughput. Plus, extended macro functions include several 'pseudo-functions' such as .LEN and .INSTR (which can be used in string substitutions). A large toolkit of useful macros is included for conditionals, loops, stack handling, and subroutines.

GST-ASM features include:

- ☐ GEM text editor (GST EDIT)
- ☐ GEM "shell" operating environment
- ☐ Unique, high level instruction macro library (IF, WHILE, REPEAT, CASE, etc.)
- ☐ Supports conditional assembly
- ☐ Creates cross-reference listings
- ☐ Symbol table
- ☐ Supports include files
- ☐ Generates relocatable code
- ☐ Produces object code compatible with: Lattice C, Pascal, Fortran, Modula 2, etc.
- ☐ Comprehensive printed user manual, including thorough documentation of all assembler error messages.

TH9017 **$59.95**

FREE WITH GSTC & GST-ASM! GST EDIT and GST LINK

GST-EDIT is a GEM-based screen editor that is to programming, what 1ST WORD is to word processing. Features up to four independent windows, cut & paste, search & replace, plus is 100% integrated with GSTC and GST-ASM.

GST LINK is the same linker that comes with Lattice C. It is an extremely flexible linker that is relocatable, supports SID symbols, symbol tables, link maps, batch control files, and more.

The standard for the 68000.

LATTICE C™

By Metocomco

- ☐ Full Kernighan and Ritchie implementation
- ☐ Powerful data types (pointers, arrays, structures, unions)
- ☐ Separate and conditional compilation
- ☐ Macros
- ☐ LATTICE design
- ☐ True native code compiler
- ☐ Comprehensive error handling, including warning messages
- ☐ Full floating point arithmetic
- ☐ Optimized to produce fast, compact code
- ☐ No runtime licenses required
- ☐ All C language features are supported, including:

PRE-PROCESSOR COMMANDS: #include, #define, #undef, #if, #ifdef, #ifndef, #else, #endif, #line.
STORAGE CLASSES: extern, static, auto, register, typedef.
TYPE DECLARATORS: int, char, short, unsigned, long, float, double, struct, union.
OBJECT MODIFIERS:, ", [ ], ( ). Declarations may be arbitrarily complex.
INITIALIZERS: Full range of expressions accepted.
STATEMENT TYPES: All are supported, including labels and goto.

- ☐ LINK 68 and GST LINK compatibility.

GEM Libraries

- ☐ Complete interface to GEM VDI and AES functions and to library of Unix and utility functions, allowing all the features of the Atari ST—icons, windows, graphics, etc.—to be used. The graphics libraries are also included in source code form.
- ☐ 270-page manual

EDITOR LINKER MENU+

TH9007 **$149.95**

MCC PASCAL™

By Metocomco

ISO Pascal Compiler

This Pascal compiler has been widely used on the Commodore Amiga and the Sinclair QL. It is a fast and powerful implementation that conforms *fully* to the exacting ISO 7185 standard. **MCC PASCAL** is the ideal Pascal for all users, whether new to Pascal or experienced programmers.

- ☐ FAST, SINGLE-PASS COMPILATION. **MCC PASCAL** compilation is straightforward and fast—ideal in education or program development.
- ☐ COMPATIBLE—COMPLETE IMPLEMENTATION OF ISO 7185. Compatibility with the International Standards Organization's definition of ISO Pascal.
- ☐ 32-BIT INTEGERS AND 32-BIT IEEE FLOATING POINT.
- ☐ COMPREHENSIVE ERROR HANDLING. The compiler recognizes over 150 different errors, and the runtime system provides over 30 different English error messages.
- ☐ 215-PAGE MANUAL.

EDITOR LINKER MENU+

TH9006 **$99.95**

"For an environment that's superb for development, I vote for Metacomco."

Anita Sinclair, Magnetic Scrolls, UK
(developers of THE PAWN)

MCC MACRO ASSEMBLER

By Metacomco

The MCC MACRO ASSEMBLER is the companion assembler for Lattice C and MCC Pascal. This full-specification assembler was specifically designed to complement those languages. Features include: linker, editor, GEM libraries, macros, Menu+ and 100-page manual.

TH9005 **$79.95**

FREE WITH EACH METACOMCO LANGUAGE
SCREEN EDITOR, LINKER, MENU+

- ☐ **EDITOR** features include: Horizontal and vertical scrolling, file merging, cut & paste, search & replace, macros.
- ☐ **LINKER** features include: Relocatable, optional SID debugger symbols, auto run-time relocation.
- ☐ **MENU+** is a remarkable new menu-driven GEM environment designed to speed up your program development cycle (EDIT, COMPILE, LINK, RUN). You can now specify a number of TOOLS (for example, any editor, compiler, debugger, etc.) together with the options for each of these tools. And MENU+ remembers all the little things that you do during a programming session—so you don't have to.

Technical questions? Talk with a Metacomco System Software Engineer: Call (408) 438-7201.

RENAME files singly or batch

CREATE and DELETE folders

MOVE files in and out of subdirectories or to other disks

TEXT FILE FORMAT and PRINT using headers, footers, and page numbers

COPY files

FORMAT disks

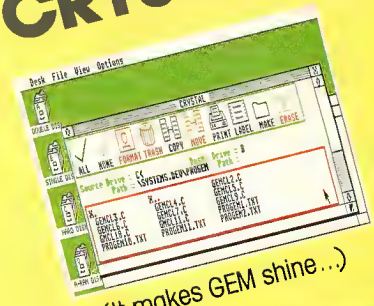
DISK LABELS printed automatically

DELETE files

NEW!

ADD A WORLD OF DOS ICONS TO ALL OF YOUR GEM PROGRAMS.

CRYSTAL™



(It makes GEM shine...)

By Jim Thompson

These are the DOS-Icons Digital Research forgot when they wrote GEM. It's too bad Atari didn't put **CRYSTAL** into the TOS ROM's, because for less than 30K of code, **CRYSTAL** gives TOS the most powerful features of MS-DOS. Plus, you never lose the look and feel of GEM. And since **CRYSTAL** is a desk accessory, it's always available.

In fact, these eight icons will eliminate all the frustration you feel from not having powerful DOS functions available from within your favorite GEM program (or the desktop).

Of course, **CRYSTAL** is compatible with all monitors, all disk drives, and all printers.

Make your personal version of GEM shine—for $24.95.

ST0229 **$24.95**

Use the language that professionals use from Prospero—the language specialist

PRO FORTRAN–77 NEW!

PRO FORTRAN–77 compiles up to 10,000 lines and utilizes over 1,000 identifiers. Pro Fortran–77 is a full implementation of ANSI (American National Standards Institute) Fortran 77 with no omissions and many useful extensions.

PRO FORTRAN–77 gives programmers a secure base on which to build.

Package includes: compiler, linker, runtime libraries, librarian, X-ref program, sample programs and 200 page manual.

Fortran is the most widely used language by universities, scientists and professional programmers. Take advantage of Fortran power combined with the wealth of information and subroutines that have been developed over the past 25 years. There is software for everything from particle physics to mechanical engineering.

Ideal for Atari software developers, schools and colleges, computer science students, training institutions and great for solving technical problems.

Fortran should be in every ST programmers library.

TH9012 **$149.95**

TOLL FREE

Credit card holders, call toll-free, 24 hours-a-day
VISA 800-443-0100, ext. 133 (Continental U.S. and Hawaii)

ORDERS ONLY!

The Pragrammers Chaice

A bit-map 'raster cutter' for ST program designers using the 16-color mode.

C.O.L.R. Object Editor

by The Rugby Circle

NOW, the first all purpose graphics design programming package is available for the ST. An essential time-saving tool developed for creating software sprites and bit-mapped game objects.

Your original art or favorite low-resolution picture from any ST paint program can be sketched or modified. Invert, twist or copy your objects. The zoom editing feature gives your picture that professional look. Your object is stored on disk in a compatible format allowing you to access your work from the programming language of your choice.

Store up to 20 objects in memory at once and instantly access four rainbow paint palettes from RAM.

C.O.L.R. OBJECT EDITOR includes object motion examples. This is the only graphics programming tool of its kind for the professional and hobbyist programmer.

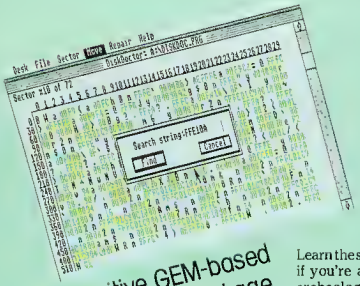
See the November 1985 issue of ANTIC for more information on **C.O.L.R. OBJECT EDITOR**.

COLOR ONLY

ST0201 **$29.95**

DISK DOCTOR

by Dan Matejka



Intuitive GEM-based disk utility package.

Learn the secrets of ST disk structure, and if you're adventurous, try some "disk archeology." Recover deleted files *automatically* and repair damaged files and disks.

☐ Make quick modifications to *any* ST single or double-sided floppy or hard disk.

☐ Use **DISK DOCTOR** to *customize* program menus and messages.

☐ Search for character strings or go to any sector instantly. Quickly edit full sectors on-screen using your mouse or cursor keys. Get directory history, file attributes, and more.

DISK DOCTOR is perfect for the casual user who just wants to repair a file, OR the *serious programmer* who wants to discover what's really going on. Includes on-disk Help and thorough technical reference information.

COLOR / MONOCHROME

ST0211 **$29.95**

TOLL FREE

Credit card holders, call toll-free, 24 hours-a-day
VISA 800-443-0100, ext. 133 (Continental U.S. and Hawaii)

ORDERS ONLY!

ULTIMA II

By Sierra

A Consistent Alter Ego Best Seller

ULTIMA II. One of the most popular fantasy role-playing games ever, **ULTIMA II** still claims a place on the bestselling charts. Second in the series this multiple disk epic that lets you create your own super-powered alter ego to perform brave and daring feats in your honor. Together, step through time doors to ages past, present and future. Explore the far reaches of the galaxy. Battle fierce monsters and cunning brigands. Finally, face the all-powerful enchantress Minax, in a duel unequaled in the annals of time. A colorful cloth map helps guide the player through the Ultima universe.

Plan on spending a couple weeks unravelling the ribald adventure.

TH9002 **$59.95**

HARD TO FIND CABLES $19.95

THE CATALOG has premium quality shielded cables with gold contacts.

What makes these cables so special? They have the unique non-standard Atari connectors and provide the length that can free-up your workstation of awkward peripheral arrangements.

520 ST
6′ ST Micro floppy disk drive cable (SF354 and SF314 drives)
PH0003 $19.95
6′ ST to printer cable
PH0001 $19.95
5′ ST to modem cable
PH0002 $19.95

800/XL/XE
10′ Atari standard peripheral cable (Serial/SIO connector)
PH0006 $19.95
5′ 850 Interface to printer cable
PH0004 $19.95
5′ 850 Interface to modem cable
PH0005 $19.95

HIPPOVISION VIDEO DIGITIZER

Create digitized Degas and NEOchrome-compatible picture files using any high quality composite video input (e.g. color or b&w camera, VCR, TV, etc.…). Fast software updates the ST's screen 10 times per second, while the hardware frame-grabber snaps pictures in 1/60 of a second. Animation mode allows automatic sequential frame digitization. Black and white high contrast mode or up to 16 grey levels with color monitors. 320H × 250V resolution. NTSC, PAL, and SECAM compatible. Includes hardware interface, software, demos and instructions.

Perfect for artists, game designers, and video aficionados.

In single color mode: It snaps the picture in a single video frame (1/60th of a second). No blurring of the picture occurs in this mode. Animate a sequence of pictures (up to a total of 50 pictures with a 512K machine, about 100 for a 1024K machine).

THE SPECS

- ☐ 256×256×9 bit resolution (3 bits per color).
- ☐ Standard NTSC 1 volt peak-to-peak composite sync signal.
- ☐ Software color "voting" system picks optimum 16-color ST palette according to image.

HIPPOPOTAMUS SOFTWARE

- ☐ Hardware plugs into printer port for high speed data transfer rate.

PH0007 **$139.95**

S/Terminal plus SOURCE!

Get on-line and transfer files with S/Terminal, a full-featured terminal program written in 68000 assembly language. S/Terminal features Xmodem, Xon/Xoff, 300/1200/2400 baud support (and more, up to 19.2K baud), and on-line help screen. ALL SOURCE CODE IS INCLUDED, in addition to object code. This disk also includes several C source and object graphic examples, plus five LOGO demos.

PD0057 **$12.00**

ST DOODLE plus SOURCE

The perfect GEM learning tool. PD paint program written in "C", including object and source files for you to explore. Works in all three resolution modes. Demonstrates GEM drop-down menus, windows, scroll bars, color selection, fill algorithm, three brush sizes. Comes with NEDVERT—converts your pictures from NED to DOODLE format. Learn how GEM and the ST work . . . without any typing!

PD0058 **$12.00**

DEGAS COMPETITION COLOR WINNERS

The top eight color entries from Batteries Included's DEGAS art competition. Includes a slide-show viewing program—DEGAS is not required.

PD0077 **$12.00**

ST FRACTALS plus SOURCE!

Features MANDLEZDDM by Harry Koons. Uses Mandelbrot algorithm to draw fractals in GEM windows in any resolution. Then zoom in with 2×, 4×, 8×, or 16× magnification. Change fractal iteration values and rescale fractals to enhance their color. Then save your fractal picture to disk. Includes all "C" source and object files. PLUS, a half dozen other fractal programs that use different algorithms and display techniques (some also with source code).

PD0068 **$12.00**

DEGAS COMPETITION HI-RES WINNERS

The top eight monochrome entries from Batteries Included's DEGAS art competition. Includes a slide-show viewing program—DEGAS is not required.

PD0076 **$12.00**

CAD-3D COLLECTION I

NEW!

The first collection of complex objects created by Tom Hudson with CAD-3D. Includes: MONITOR (Atari SM124 monitor), ALPHDM (Alpha when viewed from one angle, Omega when viewed from another), STARSHIP (Starship Enterprise), SHUTTLE (Executive Space Shuttle), OUTLET (exploded view of electrical wall outlet), HELMET (16th century Japanese battle helmet), and more! (requires CAD-3D ST0214)

PD0085 **$12.00**

ST BASIC/LOGO SAMPLER

BASIC: Includes MIDIREC.BAS—a simple MIDI sequencer and sample song files, BG.BAS—backgammon, Fractals in BASIC, Biorythm's, and more. LOGO: Nearly a dozen useful routines including complex graphics. Plus two bonus desk accessories.

PD0078 **$12.00**

SOLID SOURCE CODE

Features Jim Luczak's VDI SAMPLER and C PRIMER, which demonstrates C programming techniques and the use of VDI functions and their C BINDINGS. BICALC, a desk accessory Binary-Hexadecimal-Decimal calculator. Plus two very fast versions of LIFE, written in Assembler. All source and object code is included and is well commented. And more!

PD0079 **$12.00**

XLISP

NEW!

Interested in artificial intelligence? XLISP is a version of LISP that has an object oriented extension capability. It allows you to experiment with AI algorithms and structures as well as object oriented programming. Applications written on the ST can be ported to other computers since versions of XLISP are also running under UNIX, VAX/VMS, MD-DOS, CP/M-86, CP/M-68, and Macintosh. XLISP comes with SOURCE CODE (in C) and is extensible.

PD0084 **$12.00**

An Electronic Jewish Mother
Created by a Former Playboy Cartoonist!

Written by Yakov Kirschen and Just For You, Inc.

MOM and ME

Just think. Your own computerized Jewish mother—on-screen in an overstuffed chair; knitting, cajoling, dispensing advice, and offering you encouragement—or making you feel guilty, of course. She'll speak to you by name and brag about you to your friends.

ST0204 **$19.95**

monochrome/color

MURRAY and ME

Some people don't want a Jewish mother. How 'bout a Jewish uncle? The London Times called MURRAY "the first in a new generation of Biotoons"—computerized, interactive cartoon characters. MURRAY is always ready to cheer you up when you're feeling blue.

ST0203 **$19.95**

monochrome/color

Special Note To ST Dealers

When it comes to Atari 16 bit products, ***THE CATALOG*** is the best resource available for state-of-the art software. Every effort has been made to contact ST dealers nationwide with a copy of ***THE CATALOG*** that features the latest in ST products. If you were overlooked, it was not intentional, and we hope you will call us for free product information and dealer discounts. Be one of the first to stock your shelves with ST products from Antic, Batteries Included, Regent, Hippopotamus, GST, Kuma ...and the list continues.

Many of our customers call us looking for the nearest dealer of ST products. If you would like to be on our referral list please let us know. We're here to help you service the many ST owners...and the list is growing!

TOLL FREE

Credit card holders, call toll-free, 24 hours-a-day

VISA 800-443-0100, ext. 133 (Continental U.S. and Hawaii)

ORDERS ONLY!